BytemanPkg

V1.2.6

Installation Guide
User Guide

REF : BytemanPkg/JLP/2013/12

Author : Jean-Louis PASTUREL

Description of Successive Versions

VERSION	DATES	State	Author
V1.0	06/Oct/2013	Initial Version	JL PASTUREL
V1.1	10/Oct/2013	Adding dynamic submission	JL PASTUREL
V1.2.0	31/Oct/2013	Charting (1.0.0) + improving dynamic submission	JL PASTUREL
V1.2.1	15/Nov/2013	Improving dynamic submission(su/sudo) Correct bugs on some Helpers (Multi threading) Improving charting (V 1.1.0)	JL PASTUREL
V1.2.2	16/Nov/2013	Adding support JDK IBM to JMX remote action wit javaagent only.	JL PASTUREL
V1.2.4.1	13/Dec/2013	Integrate Byteman V 2.1.4.1 Testing with WebSphere 8.5 + OpenJDK 8. Improving this Doc	JL PASTUREL
V1.2.4.2	21/Dec/2013	Bug Dynamic submission for System Properties (Change from pkgbmsubmit.sh to pkgbminstall.sh)	JL PASTUREL
V1.2.5	18/Jan/2014	Correcting bug in byteman.jar (Submitting a method with a returnType in front), + adding -p <PORT> to pkgbmsubmit.sh bmunsubmit.sh scripts	JL PASTUREL
V1.2.5	23/Jan/2014	New ScaChart v 1.1.1 (improving scale handling in charts)	JL PASTUREL

Table des matières

Advertisement :
English is not my native language, so in the document, you will find a lot of syntax and grammar mistakes, awkwardness, difficulties to understand some sentences ...
Please let me know at jean-louis.pasturel-wrong-reply@orange.fr
(Remove -wrong-reply to the mail address)

1 Introduction

1.1 Context

Important :
This tool is based on the project JBOSS Community named Byteman. The main URL is :
https://www.jboss.org/byteman

Before playing with **BytemanPkg**, it is necessary to understand how **Byteman** runs. For this read this document :
http://downloads.jboss.org/byteman/2.1.4/ProgrammersGuide.pdf

BytemanPkg is a front-end GUI for Byteman, that constructs a Byteman Rules script file, with the help of template Rules and custom Helpers. All is packaged in a unique jar (byteman agent, properties, script rule, and mandatory helpers), and uploaded to the target servers.
The running JVM target needs to be instrumented with a javaagent, or with a dynamic install/submit and some other tips depending on the running WAS.
The tested WAS servers are (need at leat a JVM 1.6) :

- **JBOSS 7.2 / JBOSS -EAP-6.2 (*1)**
- **JOnAS 5.2.3**
- **TOMCAT 7.0.47 / 6.0.37**
- **WebSphere 8.5.5 (*2)**
- **WebLogic 12c (*3)**
- **GlassFish 4.0 (JEE7) (*4) needs JVM 1.7.**
- **Eclipse/Jetty 9.1**

I have only tested these versions of these WAS, it may run also with others versions with a JVM 1.6+.
(*1) JBOSS and JBOSS-EAP are RedHat trademarks
(*2) IBM® and WebSphere® are IBM trademarks
(*3) Oracle® WebLogic Server is an Oracle trademark
(*4) Glassfish has a dual license https://glassfish.java.net/public/CDDL+GPL.htm

I will test **Wildfly (New Open Source of JBOSS)** at the first stable version (RC or GA).

For every of these WASs, the configugation wil be detailed more further in this document

The product **BytemanPkg** is a kind of workbench that :
- is organized in projects
- has a library of Rule templates that can be extended

– has for each Rule, if necessary, a custom helper (that extends **MyHelper**) or by default **MyHelper** that extends the build-in helper : **org.jboss.byteman.rule.helper.Helper**
– generates a file script of Byteman Rules (byteman.btm)
– generates a properties file with general parameters and custom parameters for Rule/Helper
– packages the custom javagent **mybyteman.jar** in a unique jar
– uploads optionally the javaagent
– the javagent can be monitored : start/stop tracing (using IF Rule statement) , flushing the trace outputs (Custom Helper) by a JMX/RMI connection (JConsole or a script with the Attach API).

- **(new V1.1)** dynamic submission / monitoring in the "**Remote Actions**" Tab. **Only tested on Linux, not tested with others *nix or cygwin.**
- **(new V1.2.0)** Adding charting based on JfreeChart
- **Version 1.2.1 .** Improving remote submission with sudo/su on Jsch, correcting bugs with multitheading in Helpers. Charting version 1.1.0. Improving this documentation.
- **Version 1.2.2** Integrates JDK IBM with javagent and JMX monitoring (local and remote)
- **Version 1.2.4.1** Integrates Byteman 2.1.4.1. Improving this documentation.Tested with more WAS and OpenJDK 8.
- **Version 1.2.4.2** Bug Dynamic submission for System Properties (Change from pkgbmsubmit.sh to pkgbminstall.sh)
- **Version 1.2.5** Correcting a Byteman when typeReturn is before a method name. Adding -p <PORT> to pkgbmsubmit.sh and pkgbunmsubmit.shscripts
- **Version 1.2.6** New ScaChart v 1.1.1 (improving scale handling in charts)

The tool **BytemanPkg** is developed in Java 1.7 (use of JFX for GUI, Oracle JDK 7 needed for JFX, JFX will be certainly introduced with OpenJDK 8), but the byte code generated for the agent and helpers is a byte code targeted for JMV 1.6. So the JVM needed for **BytemanPkg** is a JVM 1.7, and to run javaagent (or submiting by Attach API) on the target WAS, is a JVM 1.6+.

Nota : I have tested also with OpenJDK 8 and OpenJFX 8 (the Two projects are separated), and it runs also correctly.
The 2 projects must be built and so after downloading a bunch of rpms on my Fedora 19 desktop, I succeeded.
The how-tos are in the 2 Urls below:
OpenJDK => http://hg.openjdk.java.net/jdk8/jdk8/raw-file/tip/README-builds.html#hg
OpenJFX => https://wiki.openjdk.java.net/display/OpenJFX/Building+OpenJFX

BytemanPkg uses several Java and Framework with the listed licenses :
Byteman 2.1.4.1 : LGPL regarding this document =>
https://github.com/bytemanproject/byteman/blob/master/docs/copyright.txt

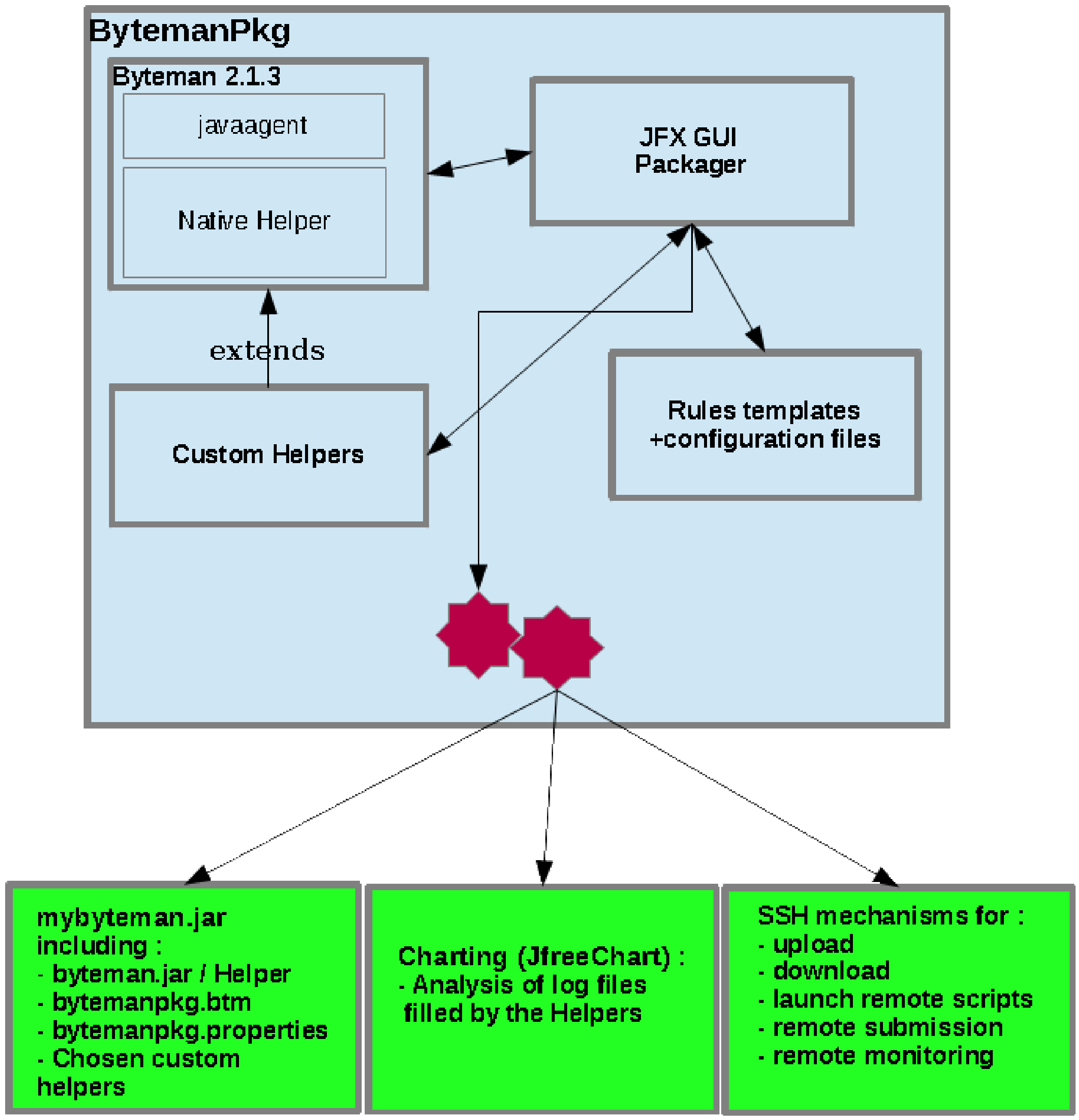
BytemanPkg Core : Apache 2 License http://www.apache.org/licenses/LICENSE-2.0.html
Jsch (BSD License) : http://www.jcraft.com/jsch/LICENSE.txt
java-sizeof : Apache 2 License http://www.apache.org/licenses/LICENSE-2.0.html
https://github.com/dweiss/java-sizeof
jfxmessagebox-1.1.0.jar : Apache, Eclipse Public License, LGPLV3

http://fr.sourceforge.jp/projects/jfxmessagebox/
Charting feature have the licence below :
Scala License : http://www.scala-lang.org/print/146
JFreeChart License : LGPL V2.1 http://www.gnu.org/licenses/lgpl-2.1.html
Akka Actors Apache 2 License : http://www.apache.org/licenses/LICENSE-2.0.html
Jcommon : LGPL V2.1 http://www.gnu.org/licenses/lgpl-2.1.html
commons-math 1.2 : Apache 2 License : http://commons.apache.org/math/license.html

The Charting feature in contained in one jar : **<bytemanPkg_Home>/lib/scaChart.jar**

1.2 Architecture

The drawing below shows, in a simplified way, how BytemanPkg runs.


BytemanPkg
Byteman 2.1.3
javaagent
Native Helper
JFX GUI Packager
extends
Custom Helpers
Rules templates +configuration files
mybyteman.jar including : - byteman.jar / Helper - bytemanpkg.btm - bytemanpkg.properties - Chosen custom helpers
Charting (JfreeChart) : - Analysis of log files filled by the Helpers
SSH mechanisms for : - upload - download - launch remote scripts - remote submission - remote monitoring

Note :
Since Byteman **Version 2.1.4.1**, the scripts rule can be acceded from the javaagent parameters in two modes :

- parameter **scripts** => a full path to the file **bytemanpkg.btm** as **scripts:/tmp/bytemanpkg.btm**
- parameters **resourcescript** => with the mean of ClassLoader .getSystemResourceAsStream("bytemanpkg.btm") as **resourcescript:jlp/byteman/helper/bytemanpkg.btm**
  this btm script is located in the archive **mybyteman.jar** (by default

-javaagent:/tmp/mybytemanJar put mybyteman.jar in the classpath of the System Classloader)

The byteman source is available in the github repo : https://github.com/bytemanproject/byteman

2 Installation

2.1 Packaging

The packaging is done in a form of zip file **bytemanPkg.zip** which the root is **bytemanPkg**

2.2 Installation of BytemanPkg

2.2.1 Requirements

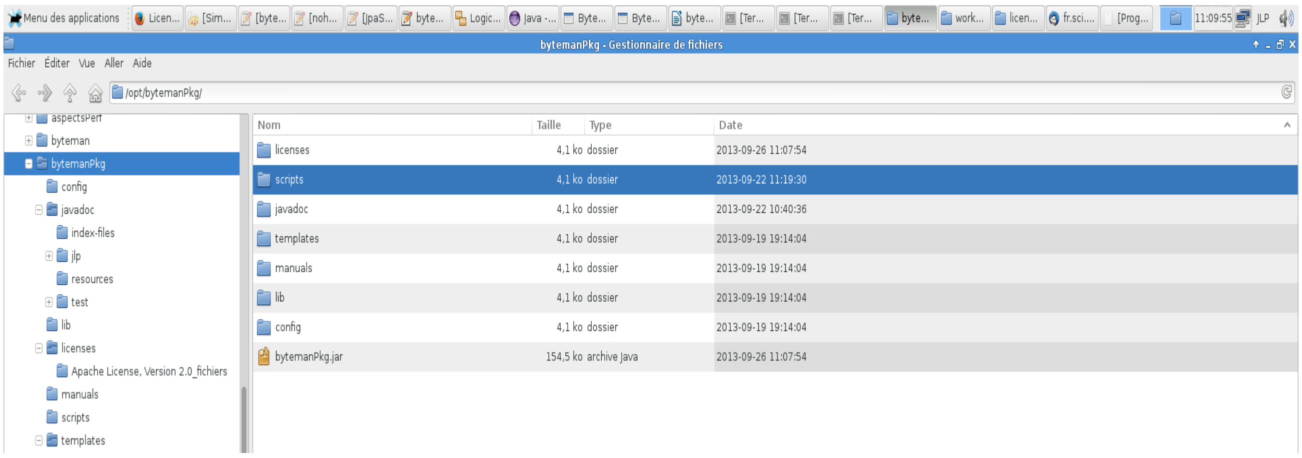
A JRE **JDK 1.7.0+ with JFX ($JRE_HOME/lib/jfxrt.jar)** must be installed on your desktop.

2.2.2 Create a deployment directory

For all the document, we suppose that you use a Linux System (there is no difficulty to adapt for a Windows System) and that the installation directory is /**opt** .
It **is not mandatory** to create the directory **bytemanPkg**.

2.2.3 De-compaction

After having downloaded **bytemanPkg.zip** in /**opt** , de-compact it in this directory.


bytemanPkg - Gestionnaire de fichiers
Fichier Éditer Vue Aller Aide
/opt/bytemanPkg/

```
aspectsPerf
byteman
bytemanPkg
    config
    javadoc
        index-files
        jlp
        resources
        test
    lib
    licenses
        Apache License, Version 2.0_fichiers
    manuals
    scripts
    templates
```

Nom	Taille	Type	Date
licenses	4,1 ko	dossier	2013-09-26 11:07:54
scripts	4,1 ko	dossier	2013-09-22 11:19:30
javadoc	4,1 ko	dossier	2013-09-22 10:40:36
templates	4,1 ko	dossier	2013-09-19 19:14:04
manuals	4,1 ko	dossier	2013-09-19 19:14:04
lib	4,1 ko	dossier	2013-09-19 19:14:04
config	4,1 ko	dossier	2013-09-19 19:14:04
bytemanPkg.jar	154,5 ko	archive Java	2013-09-26 11:07:54

2.2.4 Configuration

The configuration is set in the start script of **BytemanPkg** :
File /**opt/bytemanPkg/scripts/bytemanpkg.sh**
Make this file executable :
chmod 755 /opt/bytemanPkg/scripts/bytemanpkg.sh

```
######## These two environnement Variables must be adapted #########
###
JRE_HOME=/opt/jdk1.7.0_40/jre
## workspace directory must be created before launching thi script
workspace=/opt/eclipse/workspace/workspaceBM
#######################################################

if [[ ! -d $workspace ]]; then
       echo "the workspace : $workspace doesn't exist. Please create the directory or adapt the
variable"
       exit 1
fi

#root=/opt/bytemanPkg
root=`dirname $0`/..
echo root=$root
CLASSPATH=$root/bytemanPkg.jar
CLASSPATH=$CLASSPATH:$root/config
CLASSPATH=$CLASSPATH:$root/lib/jsch-0.1.49.jar
CLASSPATH=$CLASSPATH:$JRE_HOME/lib/jfxrt.jar
CLASSPATH=$CLASSPATH:$root/lib/antlr-runtime-4.1.jar
CLASSPATH=$CLASSPATH:$root/lib/bytemancheck.jar
CLASSPATH=$CLASSPATH:$root/lib/jfxmessagebox-1.1.0.jar
CLASSPATH=$CLASSPATH:$root/lib/scaChart.jar

$JRE_HOME/bin/java   -cp $CLASSPATH -Droot=$root -Dworkspace=$workspace -Xms128M -Xmx128M
jlp.byteman.packager.Packager
```

At the beginning of the file (in bold characters), 2 environment variables must be set, according to your installation.
This script is correct for Oracle JDK 7 and Oracle JK 8 -EA (December 2013).
For OpenJDK, you must use the script **bytemanpkgJDK8.sh** :

```
######## These some environnement Variables must be adapted #########
```

```
###
JDK_HOME=/opt/openJDK_8_X64_86

## workspace directory must be created before launching thi script
workspace=/opt/eclipse/workspace/workspaceBM

# Path to jfxrt.jar archive  not necessary  with JDK8-ea  downloaded from https://jdk8.java.net/download.html (jfxrt.jar is embedded)
# but mandatory with OpenJDK because it is a separated project => OpenJFX  built from this document :
https://wiki.openjdk.java.net/display/OpenJFX/Building+OpenJFX
JFXRT_HOME=/opt/rt/build/linux-sdk/rt/lib/ext

#########################################################

if [[ ! -d $workspace ]]; then
        echo "the workspace : $workspace doesn't exist. Please create the directory or adapt the variable"
        exit 1
fi

#root=/opt/bytemanPkg
root=`dirname $0`/..
echo root=$root
CLASSPATH=$root/bytemanPkg.jar
CLASSPATH=$CLASSPATH:$root/config
CLASSPATH=$CLASSPATH:$root/lib/jsch-0.1.49.jar
CLASSPATH=$CLASSPATH:$JFXRT_HOME/jfxrt.jar
CLASSPATH=$CLASSPATH:$root/lib/antlr-runtime-4.1.jar
CLASSPATH=$CLASSPATH:$root/lib/bytemancheck.jar
CLASSPATH=$CLASSPATH:$root/lib/jfxmessagebox-1.1.0.jar
CLASSPATH=$CLASSPATH:$root/lib/scaChart.jar
$JDK_HOME/bin/java   -classpath $CLASSPATH -Droot=$root -Dworkspace=$workspace
-Dconfig.file=$root/config/scaChart.properties -Xms128M -Xmx128M jlp.byteman.packager.Main $*
```

You must set **JFXRT_HOME/jfxrt.jar** in the classpath.

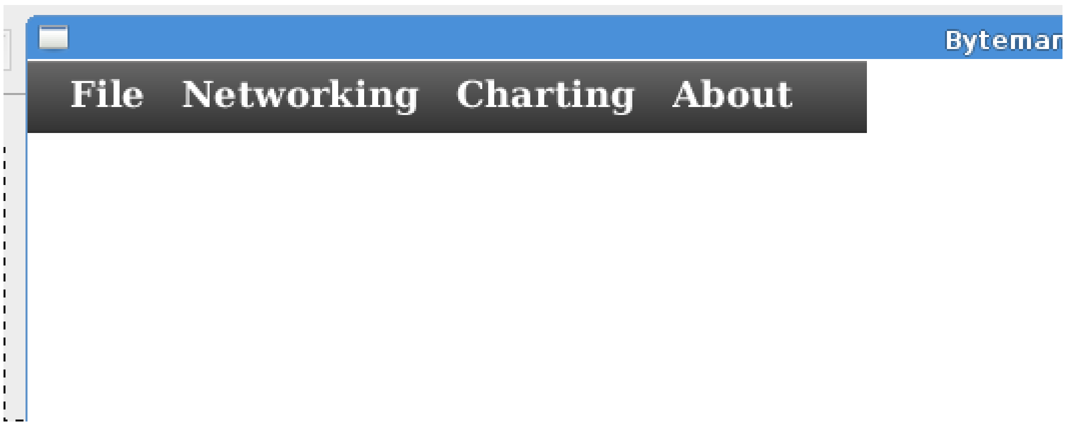
Important : The **manual creation** of the directory pointed by $**workspace** is **mandatory.**
We can after create a link on the desktop to launch **bytemanPkg** by a click on the mouse**.**

3 User Guide

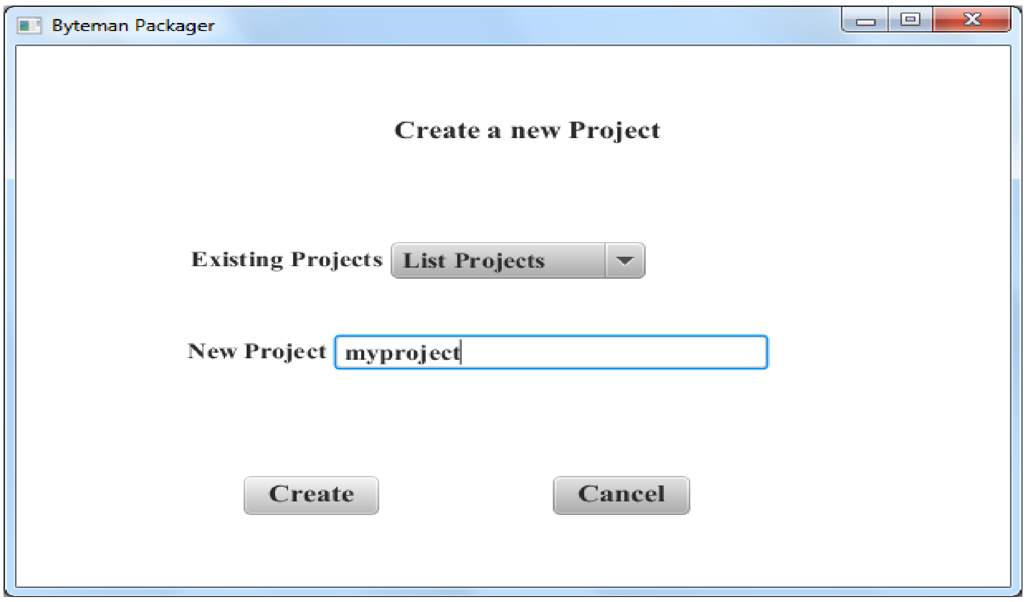
The user guide below describes step by step, how to use **bytemanPkg**.
The installation was realized as described above.

3.1 Launching BytemanPkg

By clicking the **bytemanPkg** icon on desktop, or launching the script :
<bytemanPkg_Home>/script/bytemanpkg.sh
or <bytemanPkg_Home>/script/bytemanpkgJDK8.sh with OpenJDK.

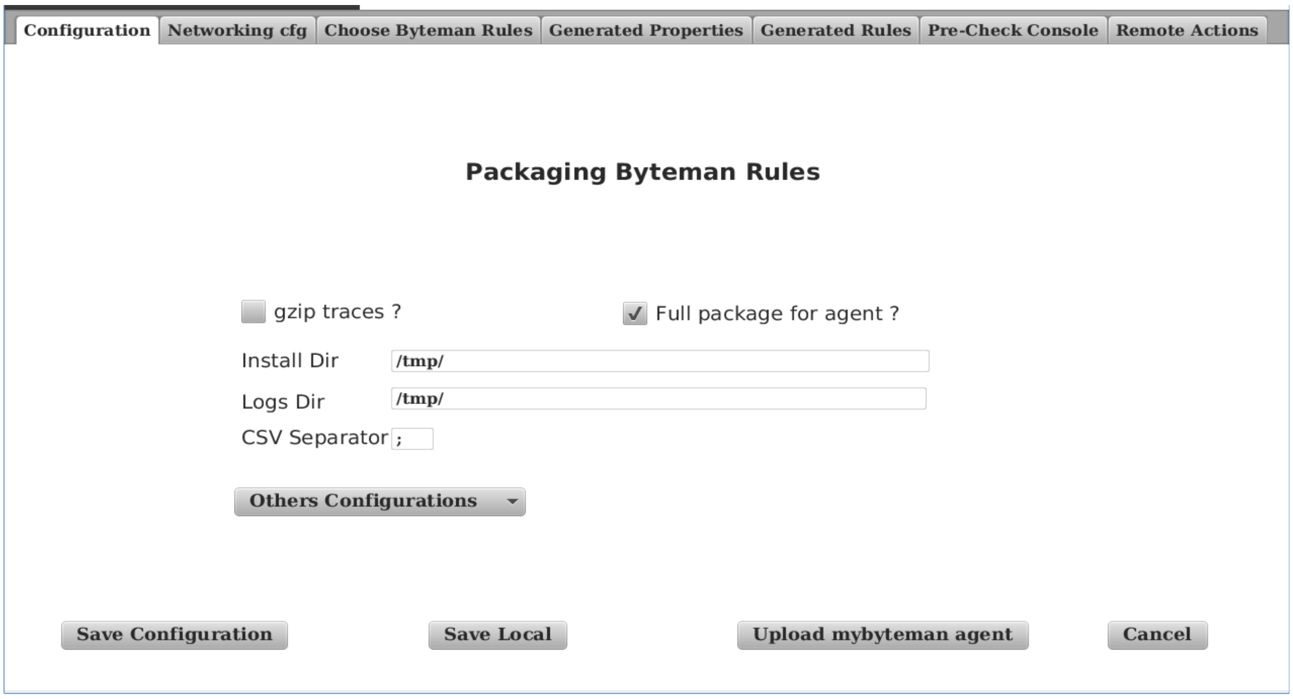


First create a project. : **myproject** for example



New Project: myproject

The screen is composed by a main tabbed pane and, at the button, a line of 4 buttons.



Packaging Byteman Rules

gzip traces ? [ ]

Full package for agent ? [✓]

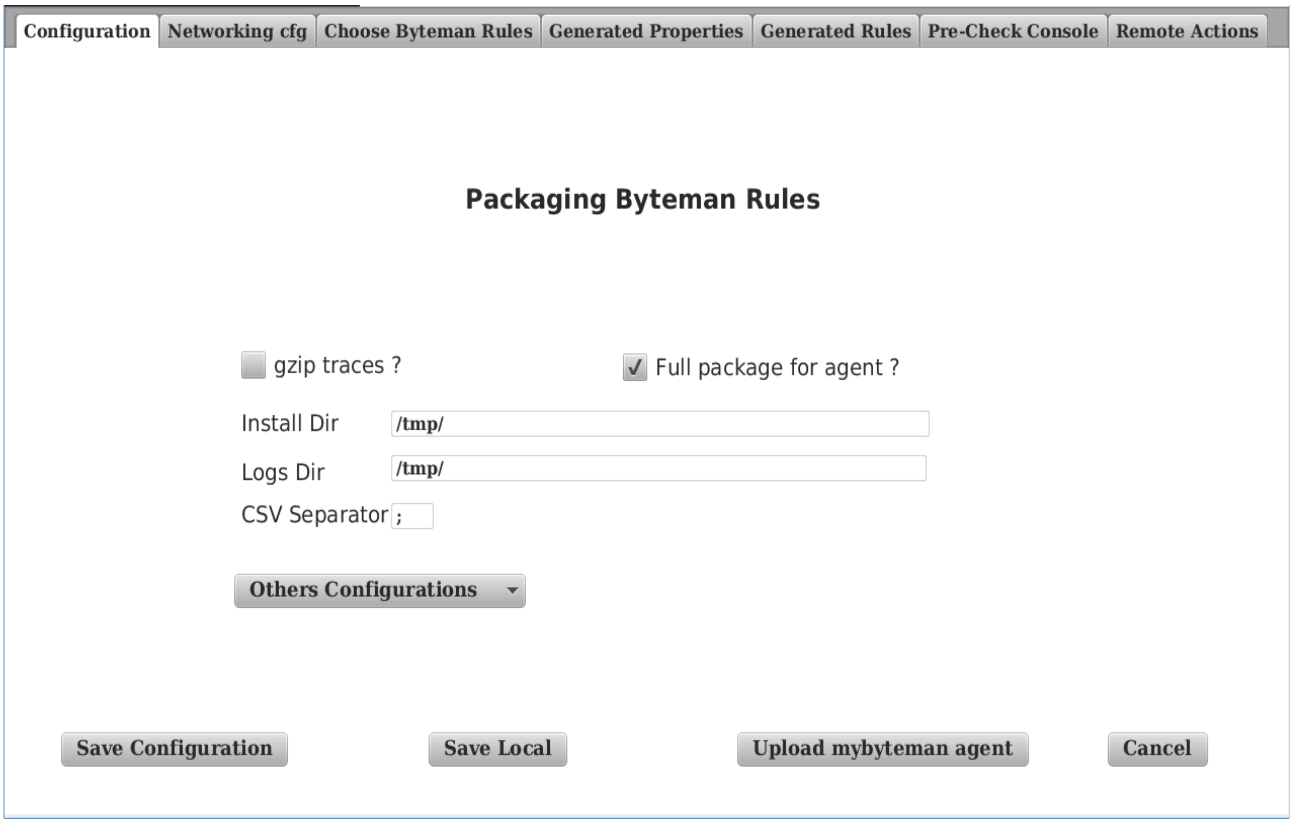
Install Dir /tmp/

Logs Dir /tmp/

CSV Separator ;

3.2 *The main tabbed Pane*

3.2.1 Configuration Tab



Packaging Byteman Rules

gzip traces ? [ ]

Full package for agent ? [✓]

Install Dir /tmp/

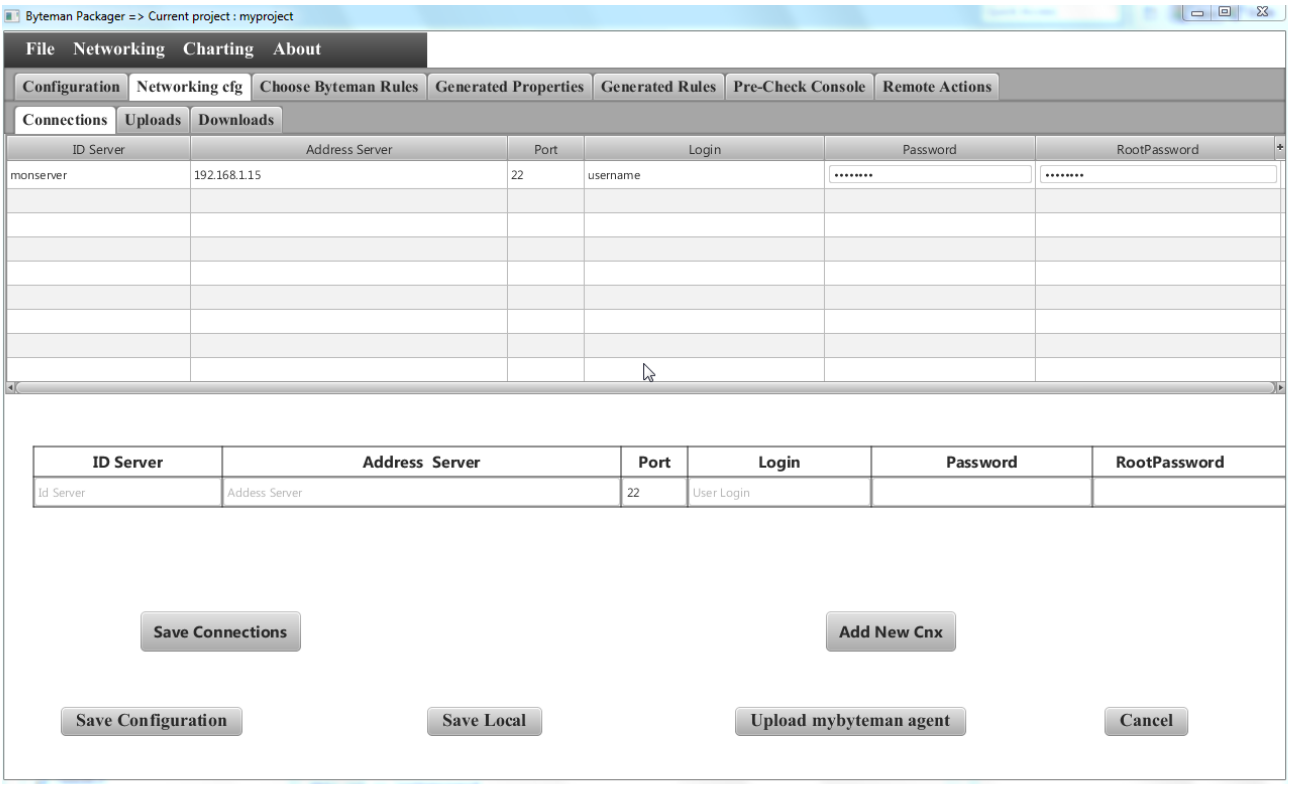
Logs Dir /tmp/

CSV Separator ;

Control	Comment
gzip	if checked trace are produced in gzip files, if not checked they are produced in plain text.
Full package for agent ?	if checked javaagent includes all Helper classes, il not checked only Helpers for chosen rules are packaged.
Install Dir	Remote directy where the javaagent must be uploaded
Logs Dir	Remote directory where trace will be produced
CSV Separator	The csv separator of the line in trace file.
Others Configurations	Shows in a ComboBox all saved locally configuration. They can be loaded.(See explanation of “Save Local' button farther.

3.2.2 Networking.cfg Tab

Connection Tab



ID Server	Address Server	Port	Login	Password	RootPassword
monserver	192.168.1.15	22	username	••••••••	••••••••

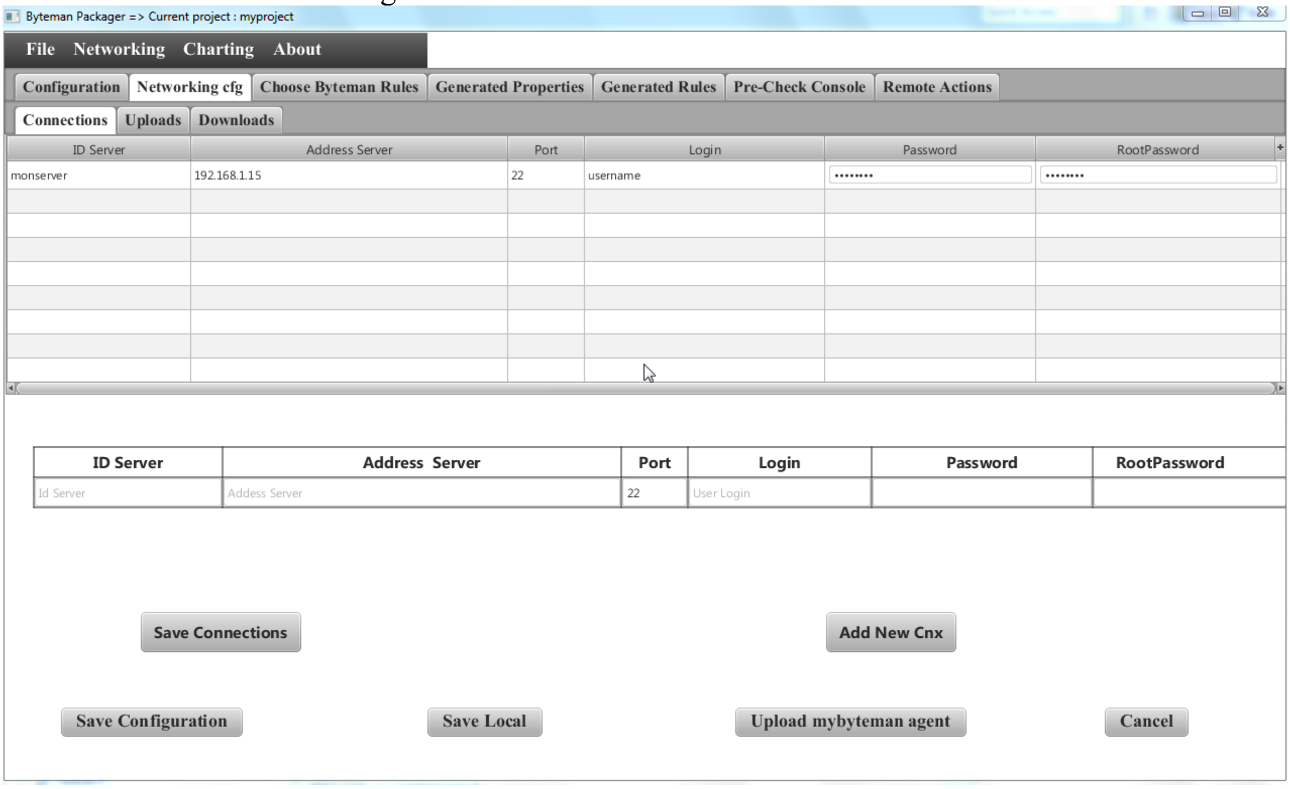
ID Server	Address Server	Port	Login	Password	RootPassword
Id Server	Address Server	22	User Login		

This Tab, is a tabbed pane with 3 tabs.

Connection Tab, defines all the remote servers of the project where the javaagent will be installed (uploaded).
To add a connection:
- fill the table with the parameters of the ssh connection
- click on "Add New Cnx"
- click on "Save Connection"

After these operations :

to delete a connection, select the line in the table and right click on it.
The fields on the View table are modifiable, you must enter a return key to valid the modification.
After a modification don't forget to save the connection.

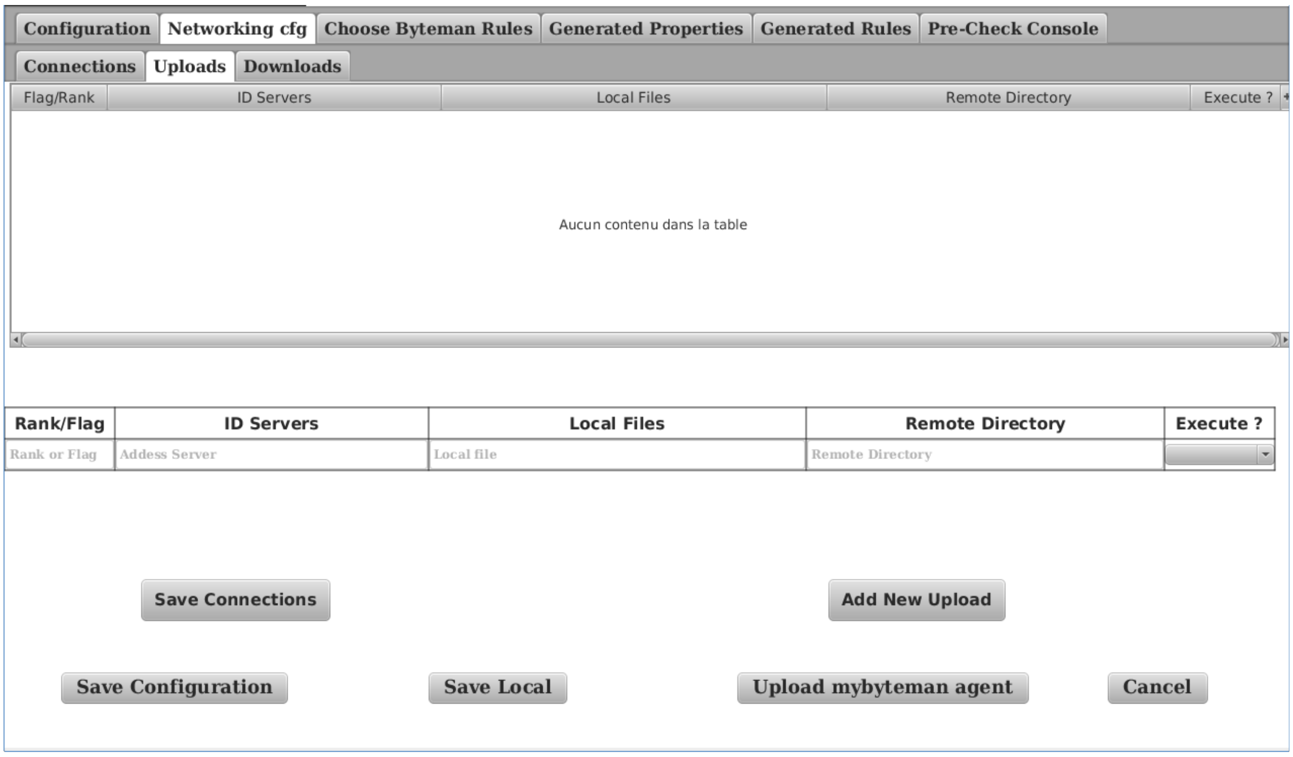


ID Server	Address Server	Port	Login	Password	RootPassword
monserver	192.168.1.15	22	username	••••••••	••••••••

ID Server	Address Server	Port	Login	Password	RootPassword
Id Server	Addess Server	22	User Login		

Upload Tab

This feature is not essential for the tool or for advanced use. It can be skipped.



This tab is used to upload files (the javagent has a dedicated mechanism with button "Upload mybyteman agent") to the remote servers. The files can be scripts that can be excecuted.
The Rank column can take 3 kind of values (integer):
-1 => means that the line is ignored (like a comment)
0 => means that the line is used in the "Uploads only" mechanism (see menu Networking/"Uploads Only")
>0 => means that the line belongs to chain of commands that starts to 1. This chain can mix uploads/execute remote Shell/downloads of result files.

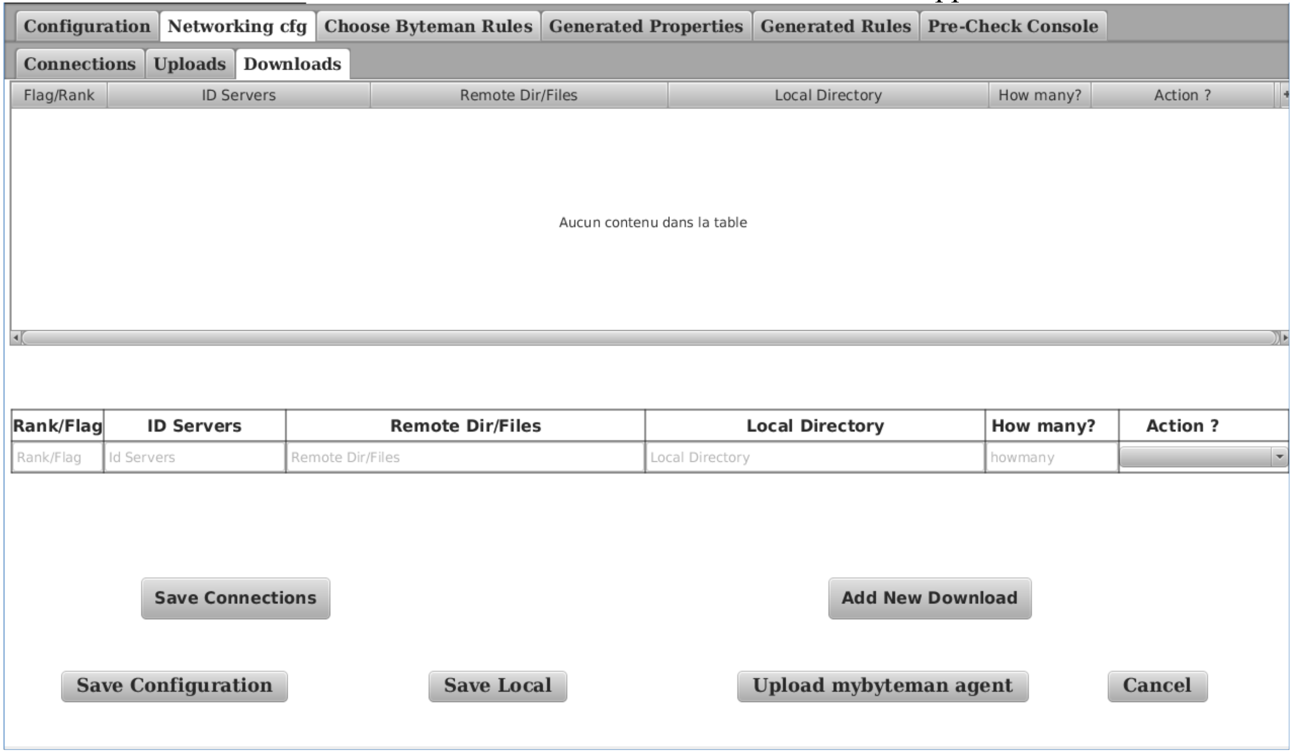
The mechanism of filling the uploads line is the same as described above for connection.

Column	Comment
Rank	Integer : -1 skip the line; 0 for Uploads Only; >0 belongs to a chain of uploads/execute/downloads see just above more explanations
ID Servers	List of servers listed in the connection tab. Joker character * is allowed like myserv*, or list of servers separated by ; .
Local File	Path to the local file
Remote Directory	Path to the remote directory. The last character must be the File Separator (/).

Column	Comment
execute	Yes => the file is set executable and executed after upload.

Download Tab

This feature is not essential for the tool or for advanced use. It can be skipped.



Flag/Rank	ID Servers	Remote Dir/Files	Local Directory	How many?	Action ?

Aucun contenu dans la table

Rank/Flag	ID Servers	Remote Dir/Files	Local Directory	How many?	Action ?
Rank/Flag	Id Servers	Remote Dir/Files	Local Directory	howmany	

This tab is used to dowload files from the remote servers.
The Rank column can take 3 kind of values (integer):
-1 => means that the line is ignored (like a comment)
0 => means that the line is used in the "Downloads only" mechanism (see menu Networking/"Downloads Only")
>0 => means that the line belongs to chain of commands that starts to 1. This chain can mix uploads/execute remote Shell/downloads of result files.

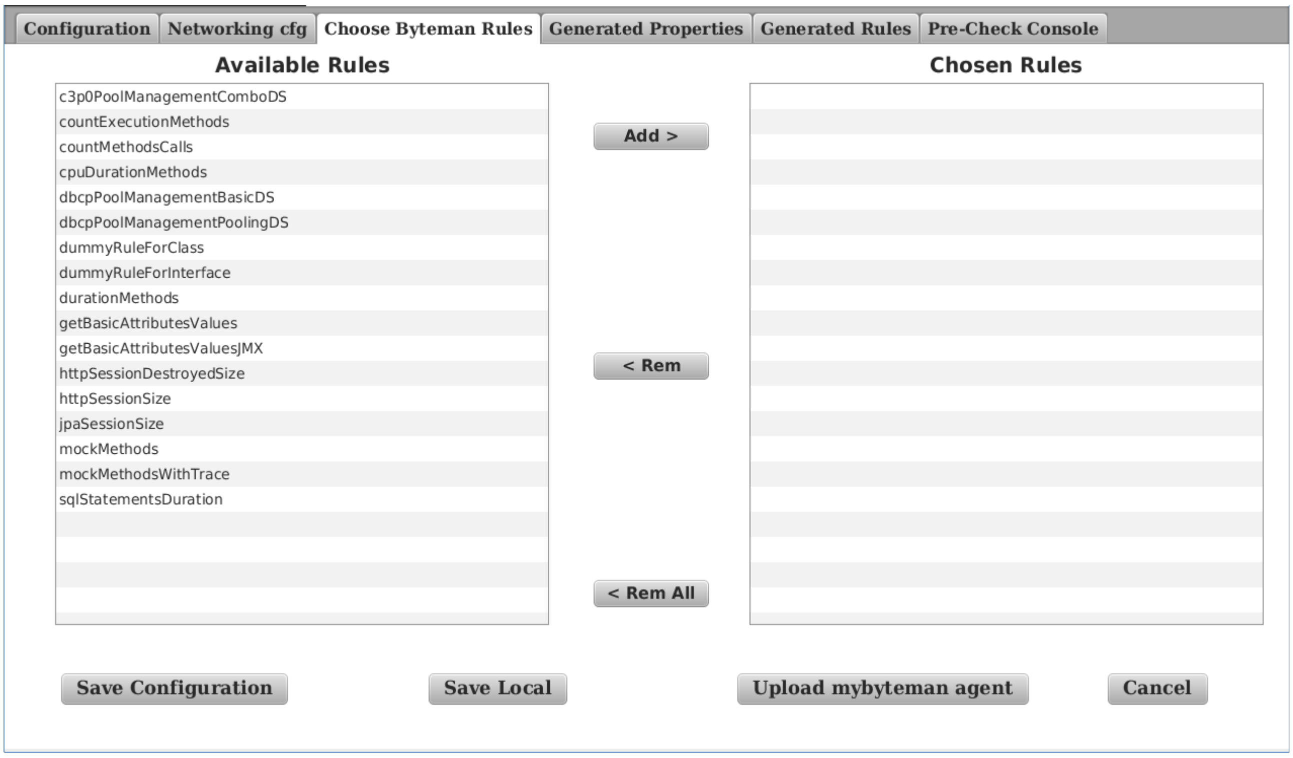
The mechanism of filling the uploads line is the same as described above for connection.

Column	Comment
Rank	Integer : -1 skip the line; 0 for Uploads Only; >0 belongs to a chain of uploads/execute/downloads see just above more explanations
ID Servers	List of servers listed in the connection tab. Joker character * is allowed like myserv*, or list of servers separated by ; .

Column	Comment
Remote Dir/Files	Path to the remote file or remote directory .For a directory, the last character must be the file separator (/). Joker character * is allowed.
Local Directory	Path to thelast directory. The last character must be the File Separator (/ or \)
How Many ?	An int, say n, that allows to download the n younger files or the n younger directories.
Action ?	**No_Compress_Prefix** => no compression before download, the local file or the local directory is prefixed by the name of the server (to avoid overwriting) **Compress_Prefix** => compression before download, the local file is prefixed by the name of the server (to avoid overwriting) **Compress_NoPrefix** => compression before download, the local file/directory is not prefixed by the name of the server **No_Compress_NoPrefix =>** no compression before download, the local file is not prefixed by the name of the server

Choose Byteman Rules Tab

It is the main tab of the tool, it packages the rules in a unique script file **bytemanpkg.btm**, and package the need helpers for the rules.

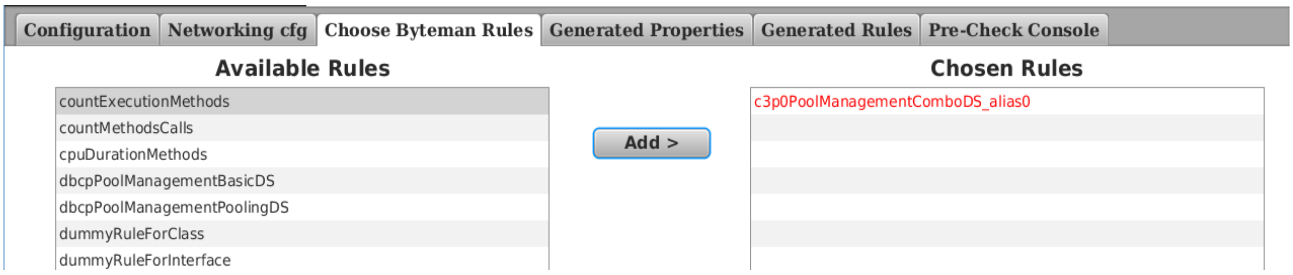


Available Rules

- c3p0PoolManagementComboDS
- countExecutionMethods
- countMethodsCalls
- cpuDurationMethods
- dbcpPoolManagementBasicDS
- dbcpPoolManagementPoolingDS
- dummyRuleForClass
- dummyRuleForInterface
- durationMethods
- getBasicAttributesValues
- getBasicAttributesValuesJMX
- httpSessionDestroyedSize
- httpSessionSize
- jpaSessionSize
- mockMethods
- mockMethodsWithTrace
- sqlStatementsDuration

Chosen Rules

The left ListView shows all the available rules. You add a rule by selecting it in the left ListView and clicking on the button :



Available Rules

- countExecutionMethods
- countMethodsCalls
- cpuDurationMethods
- dbcpPoolManagementBasicDS
- dbcpPoolManagementPoolingDS
- dummyRuleForClass
- dummyRuleForInterface

Chosen Rules

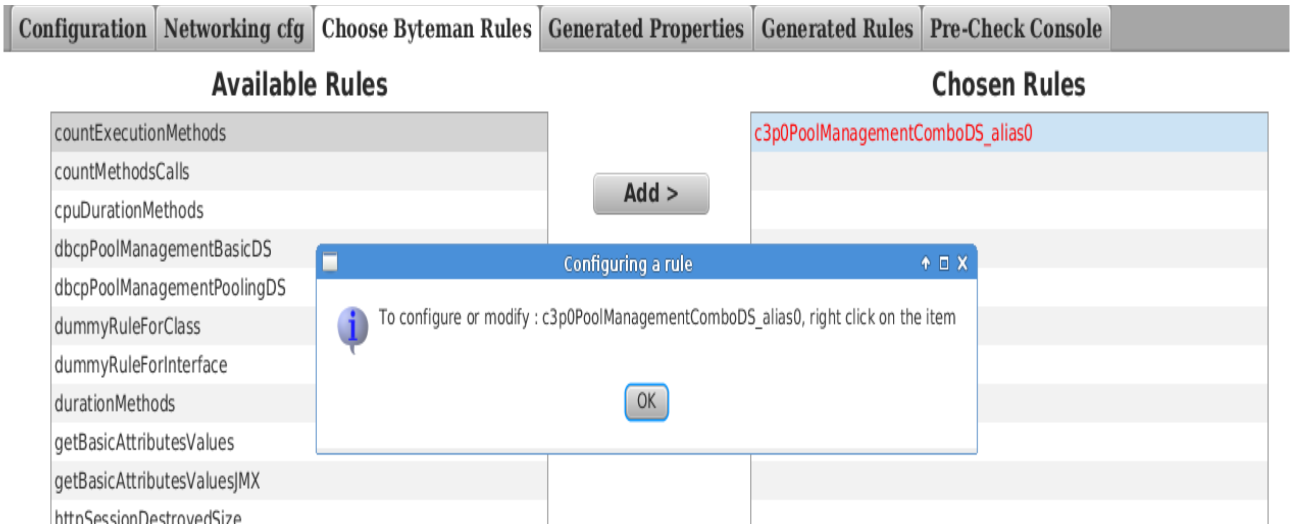
- c3p0PoolManagementComboDS_alias0

The rule in the right ListView, the first time appears in **red,** that means it is not yet configured.
Each rule must be configured.
There are two kinds of rules :
- rules that can be instantiated only once
- rules that can be instantiated several times (there are distinguished in the right ListView by the suffix _alias<n>)

For the rule that are instantiated once, the suffix _alias0 is added also in the name of rule in the right

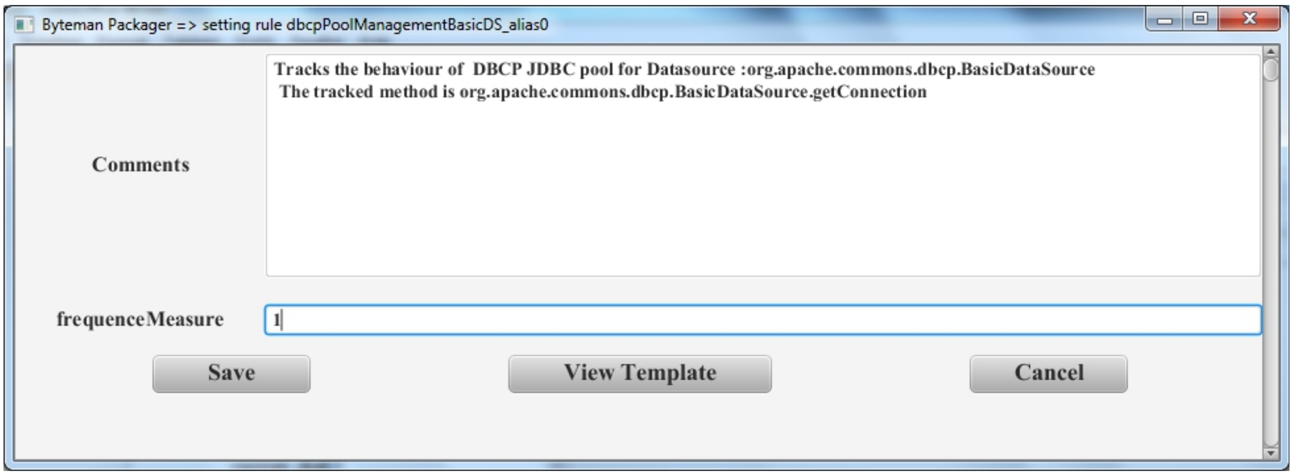
ListView. When a such rule is chosen, it disappears from the left ListView.

To configure or modify the rule, in the right ListView, **you have to right click** on the rule.



To configure or modify : c3p0PoolManagementComboDS_alias0, right click on the item

The custom dialog window appears to configure the rule. This dialog is different for each rule.


Byteman Packager => setting rule dbcpPoolManagementBasicDS_alias0
Save
View Template
Cancel

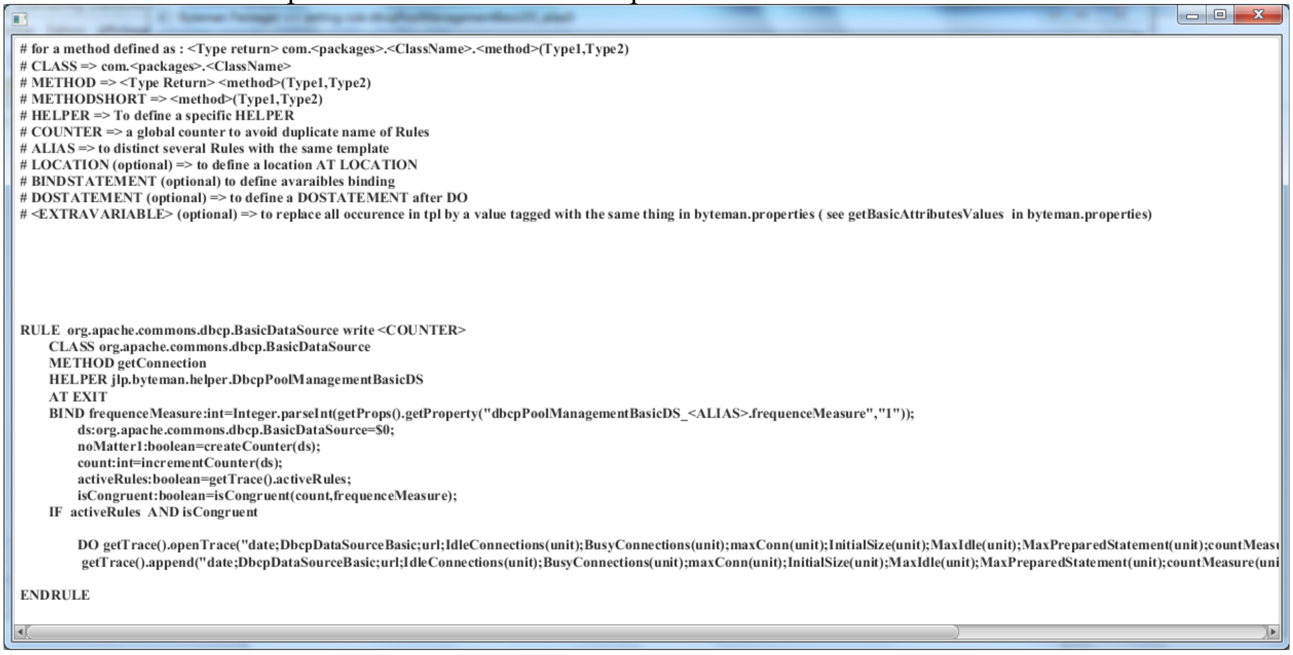
Comments
```
Tracks the behaviour of  DBCP JDBC pool for Datasource :org.apache.commons.dbcp.BasicDataSource
 The tracked method is org.apache.commons.dbcp.BasicDataSource.getConnection
```

frequenceMeasure: 1

The textArea comment explains what the rule does, and how to fill every field of the configuration. Some more complex rules are also explained in the Annexe of this document.

For curious, the rules are defined in the file :
<**bytemanPkg**_home>/config/**bytemanRules.properties**
and the templates for each rule are in the directory: <**bytemanPkg**_home>/templates/byteman/

The button View Template button shows the template used:

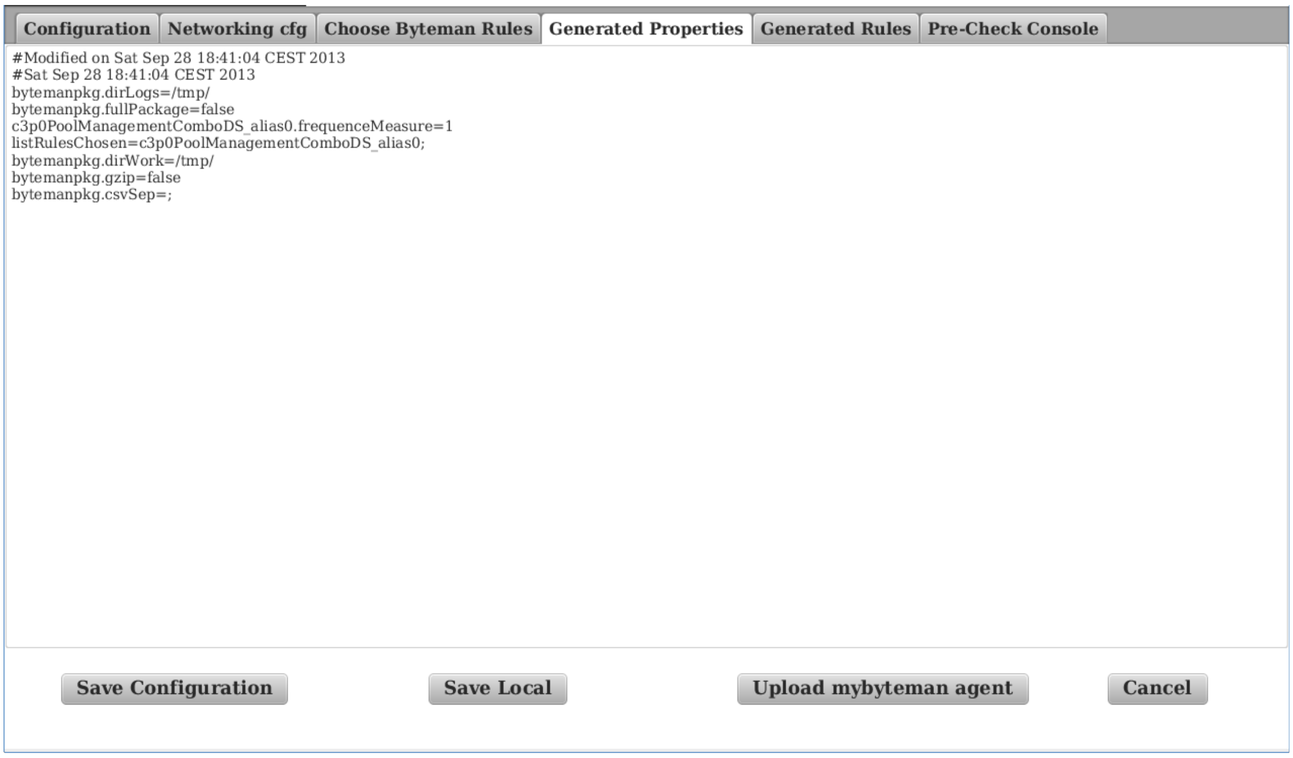


```
# for a method defined as : <Type return> com.<packages>.<ClassName>.<method>(Type1,Type2)
# CLASS => com.<packages>.<ClassName>
# METHOD => <Type Return> <method>(Type1,Type2)
# METHODSHORT => <method>(Type1,Type2)
# HELPER => To define a specific HELPER
# COUNTER => a global counter to avoid duplicate name of Rules
# ALIAS => to distinct several Rules with the same template
# LOCATION (optional) => to define a location AT LOCATION
# BINDSTATEMENT (optional) to define avaraibles binding
# DOSTATEMENT (optional) => to define a DOSTATEMENT after DO
# <EXTRAVARIABLE> (optional) => to replace all occurence in tpl by a value tagged with the same thing in byteman.properties ( see getBasicAttributesValues  in byteman.properties)




RULE  org.apache.commons.dbcp.BasicDataSource write <COUNTER>
      CLASS org.apache.commons.dbcp.BasicDataSource
      METHOD getConnection
      HELPER jlp.byteman.helper.DbcpPoolManagementBasicDS
      AT EXIT
      BIND frequenceMeasure:int=Integer.parseInt(getProps().getProperty("dbcpPoolManagementBasicDS_<ALIAS>.frequenceMeasure","1"));
            ds:org.apache.commons.dbcp.BasicDataSource=$0;
            noMatter1:boolean=createCounter(ds);
            count:int=incrementCounter(ds);
            activeRules:boolean=getTrace().activeRules;
            isCongruent:boolean=isCongruent(count,frequenceMeasure);
      IF  activeRules  AND isCongruent

            DO getTrace().openTrace("date;DbcpDataSourceBasic;url;IdleConnections(unit);BusyConnections(unit);maxConn(unit);InitialSize(unit);MaxIdle(unit);MaxPreparedStatement(unit);countMeasu
             getTrace().append("date;DbcpDataSourceBasic;url;IdleConnections(unit);BusyConnections(unit);maxConn(unit);InitialSize(unit);MaxIdle(unit);MaxPreparedStatement(unit);countMeasure(uni

ENDRULE
```

When the fields are filled click on “Save” button, and “Save Configuration” in the main screen.

Generated Properties Tab

It is only a read only TextArea tab.

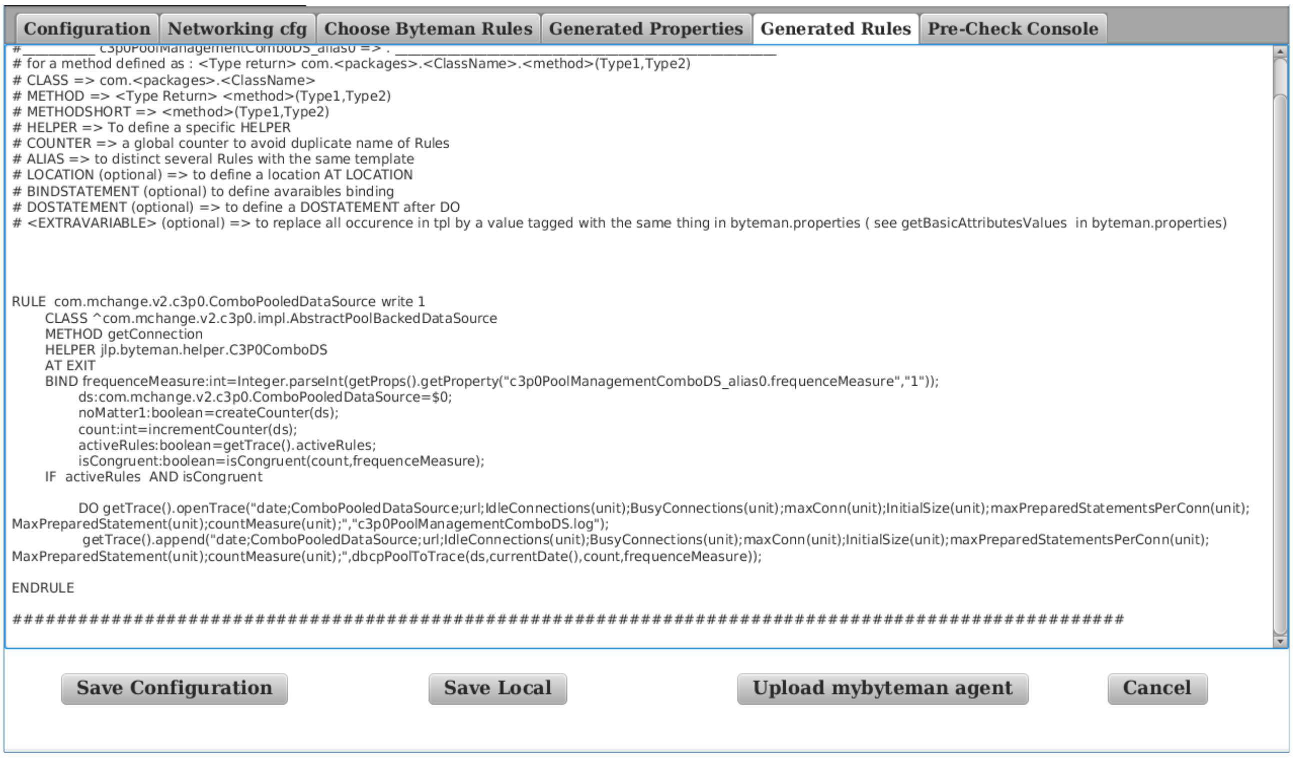


```
#Modified on Sat Sep 28 18:41:04 CEST 2013
#Sat Sep 28 18:41:04 CEST 2013
bytemanpkg.dirLogs=/tmp/
bytemanpkg.fullPackage=false
c3p0PoolManagementComboDS_alias0.frequenceMeasure=1
listRulesChosen=c3p0PoolManagementComboDS_alias0;
bytemanpkg.dirWork=/tmp/
bytemanpkg.gzip=false
bytemanpkg.csvSep=;
```

These properties are, in the file **bytemanpkg.properties,** packaged in the archive **mybyteman.jar**.

Generated Rules Tab

It is only a read only TextArea tab.

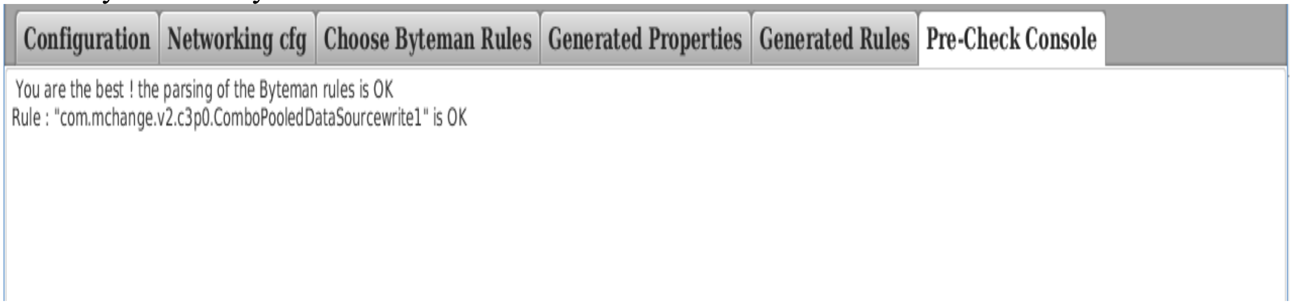


```
#__________ c3p0PoolManagementComboDS_alias0 => : ______________________________________________
# for a method defined as : <Type return> com.<packages>.<ClassName>.<method>(Type1,Type2)
# CLASS => com.<packages>.<ClassName>
# METHOD => <Type Return> <method>(Type1,Type2)
# METHODSHORT => <method>(Type1,Type2)
# HELPER => To define a specific HELPER
# COUNTER => a global counter to avoid duplicate name of Rules
# ALIAS => to distinct several Rules with the same template
# LOCATION (optional) => to define a location AT LOCATION
# BINDSTATEMENT (optional) to define avaraibles binding
# DOSTATEMENT (optional) => to define a DOSTATEMENT after DO
# <EXTRAVARIABLE> (optional) => to replace all occurence in tpl by a value tagged with the same thing in byteman.properties ( see getBasicAttributesValues  in byteman.properties)



RULE  com.mchange.v2.c3p0.ComboPooledDataSource write 1
      CLASS ^com.mchange.v2.c3p0.impl.AbstractPoolBackedDataSource
      METHOD getConnection
      HELPER jlp.byteman.helper.C3P0ComboDS
      AT EXIT
      BIND frequenceMeasure:int=Integer.parseInt(getProps().getProperty("c3p0PoolManagementComboDS_alias0.frequenceMeasure","1"));
            ds:com.mchange.v2.c3p0.ComboPooledDataSource=$0;
            noMatter1:boolean=createCounter(ds);
            count:int=incrementCounter(ds);
            activeRules:boolean=getTrace().activeRules;
            isCongruent:boolean=isCongruent(count,frequenceMeasure);
      IF  activeRules  AND isCongruent

            DO getTrace().openTrace("date;ComboPooledDataSource;url;IdleConnections(unit);BusyConnections(unit);maxConn(unit);InitialSize(unit);maxPreparedStatementsPerConn(unit);
MaxPreparedStatement(unit);countMeasure(unit);","c3p0PoolManagementComboDS.log");
             getTrace().append("date;ComboPooledDataSource;url;IdleConnections(unit);BusyConnections(unit);maxConn(unit);InitialSize(unit);maxPreparedStatementsPerConn(unit);
MaxPreparedStatement(unit);countMeasure(unit);",dbcpPoolToTrace(ds,currentDate(),count,frequenceMeasure));

ENDRULE

##########################################################################################
```

These rules are in the file **bytemanpkg.btm** packaged in the archive **mybyteman.jar**.

Pre-check Console Tab

It is only a read only TextArea tab.



```
You are the best ! the parsing of the Byteman rules is OK
Rule : "com.mchange.v2.c3p0.ComboPooledDataSourcewrite1" is OK
```

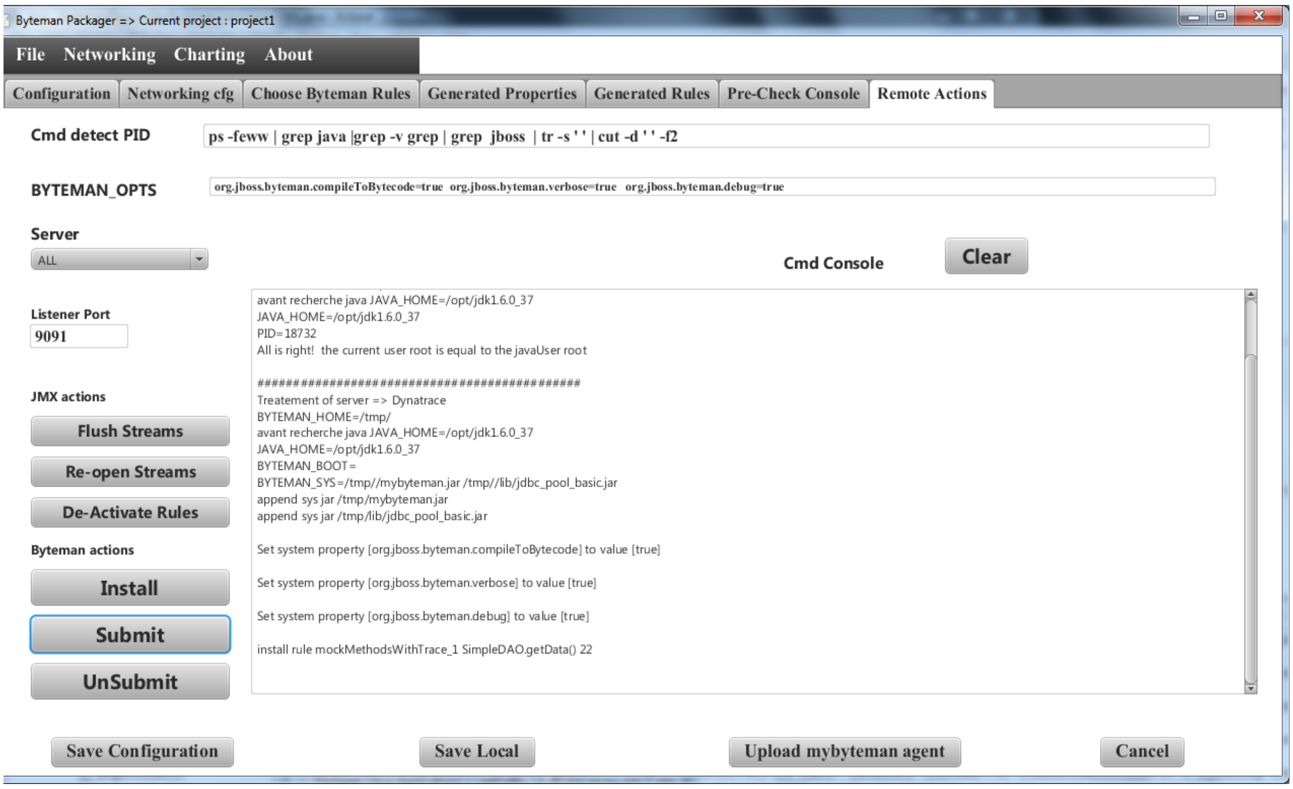
It is an experimental feature, to check if rules have a correct syntax. It is a static control, not controlled against the target code.
The used Antlr grammar was simple, and certainly doesn't cover all the syntax of Byteman rules, so it can happen false positive and false negative checks of rules.

Remote Actions Tab

(new in V1.1) Only tested on Linux, not tested with others *nix or cygwin.
Important :

> **This mechanism uses the Attach API, so Attach API must be allowed => this is the default.**
> **For HotSpot JVMs the parameter is : -XX:-DisableAttachMechanism (-XX:+DisableAttachMechanism disables Attach API).**
> **For IBM JVM the parameter is : -Dcom.ibm.tools.attach.enable=yes (-Dcom.ibm.tools.attach.enable=no disables Attach API). With IBM JVM only JMX Actions are available => see JVM IBM issue in the Annex paragraph.**

By this way, there is **no javaagent** to configure as JVM parameter. All the necessary script, jars, scripts rules are automatically uploaded on remote server and excecuted.
The only thing, we have to do, it is to allow the byte code injection for the specific WAS (see further in this document the configurations for **the tested WAS**).


Byteman Packager => Current project : project1
File Networking Charting About
Configuration Networking cfg Choose Byteman Rules Generated Properties Generated Rules Pre-Check Console Remote Actions
Clear
JMX actions
Flush Streams
Re-open Streams
De-Activate Rules
Byteman actions
Install
Submit
UnSubmit
Save Configuration
Save Local
Upload mybyteman agent
Cancel

Cmd detect PID: `ps -feww | grep java |grep -v grep | grep  jboss  | tr -s ' ' | cut -d ' ' -f2`

BYTEMAN_OPTS: `org.jboss.byteman.compileToBytecode=true  org.jboss.byteman.verbose=true   org.jboss.byteman.debug=true`

Server: ALL

Listener Port: 9091

Cmd Console

```
avant recherche java JAVA_HOME=/opt/jdk1.6.0_37
JAVA_HOME=/opt/jdk1.6.0_37
PID=18732
All is right!  the current user root is equal to the javaUser root

############################################
Treatement of server => Dynatrace
BYTEMAN_HOME=/tmp/
avant recherche java JAVA_HOME=/opt/jdk1.6.0_37
JAVA_HOME=/opt/jdk1.6.0_37
BYTEMAN_BOOT=
BYTEMAN_SYS=/tmp//mybyteman.jar /tmp//lib/jdbc_pool_basic.jar
append sys jar /tmp/mybyteman.jar
append sys jar /tmp/lib/jdbc_pool_basic.jar

Set system property [org.jboss.byteman.compileToBytecode] to value [true]

Set system property [org.jboss.byteman.verbose] to value [true]

Set system property [org.jboss.byteman.debug] to value [true]

install rule mockMethodsWithTrace_1 SimpleDAO.getData() 22
```

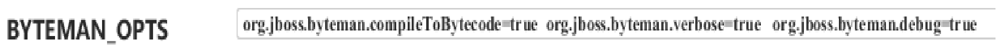
This tab allows the remote actions on byteman rules, contained in the file **bytemanpkg.btm** :
- installation
- submit / unsubmit
- JMX actions (activate / deactivate rules, re-open output streams, flush output streams)

The sh scripts, jar archives, bytemanpkg.btm files are first uploaded on the remote servers using Jsch API . The actions are launched from the client using also Jsch API.

Under **<workspace>/<current_project>** directory, there are the two folders below :
- **sys** => put in it necessary application jars, that are mandatory for rules in system classpath (generally it is the right place for application jars)
- **boot** =>   put in it necessary application jars, that are mandatory for rules in boot classpath.

The content of this 2 folders is uploaded to the remote servers.
It is the simpliest way to do that, because every WAS has its own behaviour to expand ear and war archives.

Control	Comment
Cmd Detect PID	TextField : A command that returns the PID of the java process to instrument. Based on jps (JDK/bin/jps script). A regular expression for grep is needed to return only one PID.
Server	ComboBox : ALL : All the servers declared in the connexion tab are treated, or you can choose only one server.
Cmd Console	TextArea : The stdout/stderr of the remote servers

Control	Comment
BYTEMAN_OPTS	To add byteman (submit/unsubmit) system properties (see Byteman documentation). **Important :** the install script (pkgbminstall.sh) is launched with the the setting of parameter **-Dorg.jboss.byteman.allow.config.updates=true** If it doesn't run with your JVM, try by setting this parameter directly on the targeted JVM (like with the classic javaagent). For **bytemanPkg**, you must fill the properties with a value (even if the value is not used), with no space after and before the sign = The more used are : **org.jboss.byteman.compileToBytecode=true** **org.jboss.byteman.skip.overriding.rules=true** **org.jboss.byteman.transform.all=true** **org.jboss.byteman.verbose=true** **org.jboss.byteman.debug=true** **org.jboss.byteman.dump.generated.classes=true** The last three properties are useful for debugging. The first and second parameter give better performance. The third allows to trigger also methods the Java core classes (java.*, javax.*) **Don't set -D** prefix for the system properties. An example shown below : BYTEMAN_OPTS org.jboss.byteman.compileToBytecode=true org.jboss.byteman.verbose=true org.jboss.byteman.debug=true The script **pkgbmunisntall.sh** delete all properties.
Clear	Button : to clear the cmd Console
Listener Port	TextField : the port used by the byteman agent.
JMX Action	See below Monitoring with Jconsole : - Flush Stream => operation **flushAllOs** **-** Re-open Streams => operation **reOpenOs** **-** Activate / De-Activate Rules => operations **enableRules/disableRules**
Byteman Actions	The actions correspond to the byteman scripts (install/submit) - **Install :** install the byteman agent using the Listener Port and the PID as described above - **Submit** : submit the rules contained in file **bytemanpkg.btm** - **UnSubmit** : un-submit the rules contained in file **bytemanpkg.btm**

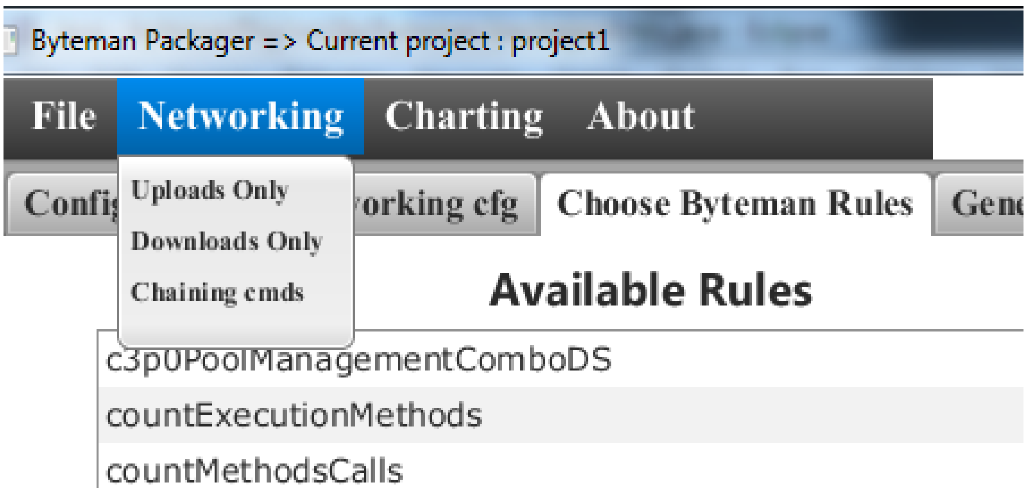
Quickly, the normal usage is :

1. **Open or Create a** project
2. **Fill** Connection Tab
3. **Fill** the folders **boot / sys** with jars if necessary
4. **Configure** Byteman Rules
5. **Launch the WAS** on remote servers

6. **Install**
7. **Submit**

Nota : the tested cases, on remote actions using su / sudo , are described here
If the remote launching fails, the solution is to connect on the targeted servers and to launch locally the scripts with the correct user.

3.3 The Menu : Netwoking



3.3.1 Uploads Only

Linked with the tab => Networking.cfg/Uploads
All the lines, where Flag/Rank equals 0 are executed (uploads files to remote servers).

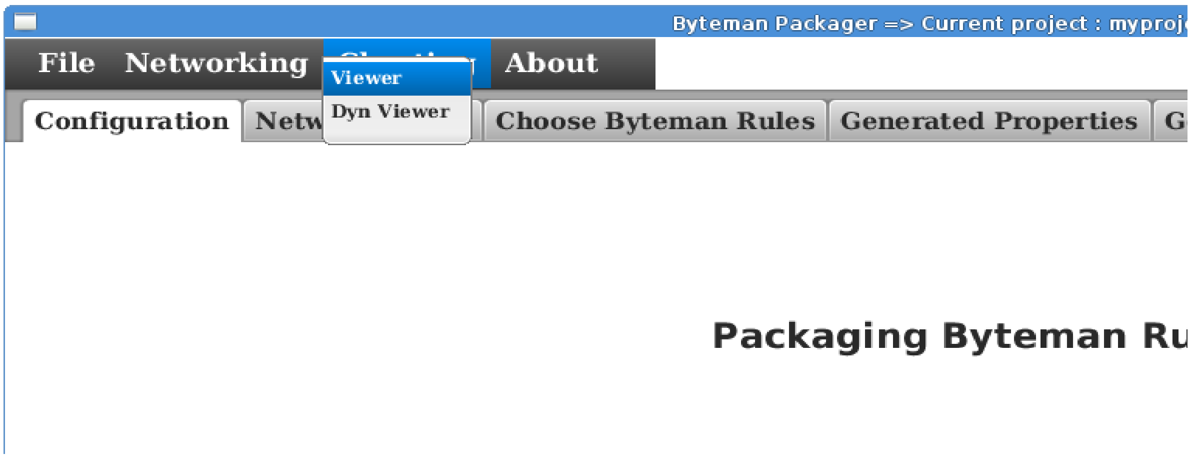
3.3.2 Downloads Only

Linked with the tab => Networking.cfg/Downloads
All the lines, where Flag/Rank equals 0 are executed (downloads files/directories from remote servers).

3.3.3 Chaining cmds

Linked both with the two tabs => Networking.cfg/Uploads and Networking.cfg/Downloads.
All the lines where Flag/Rank >0 are executed in sequence, begining by 1 with no hole in the numerotation.
You can mix uploads/remote scripts executions/downloads.

3.4 The Menu : Charting



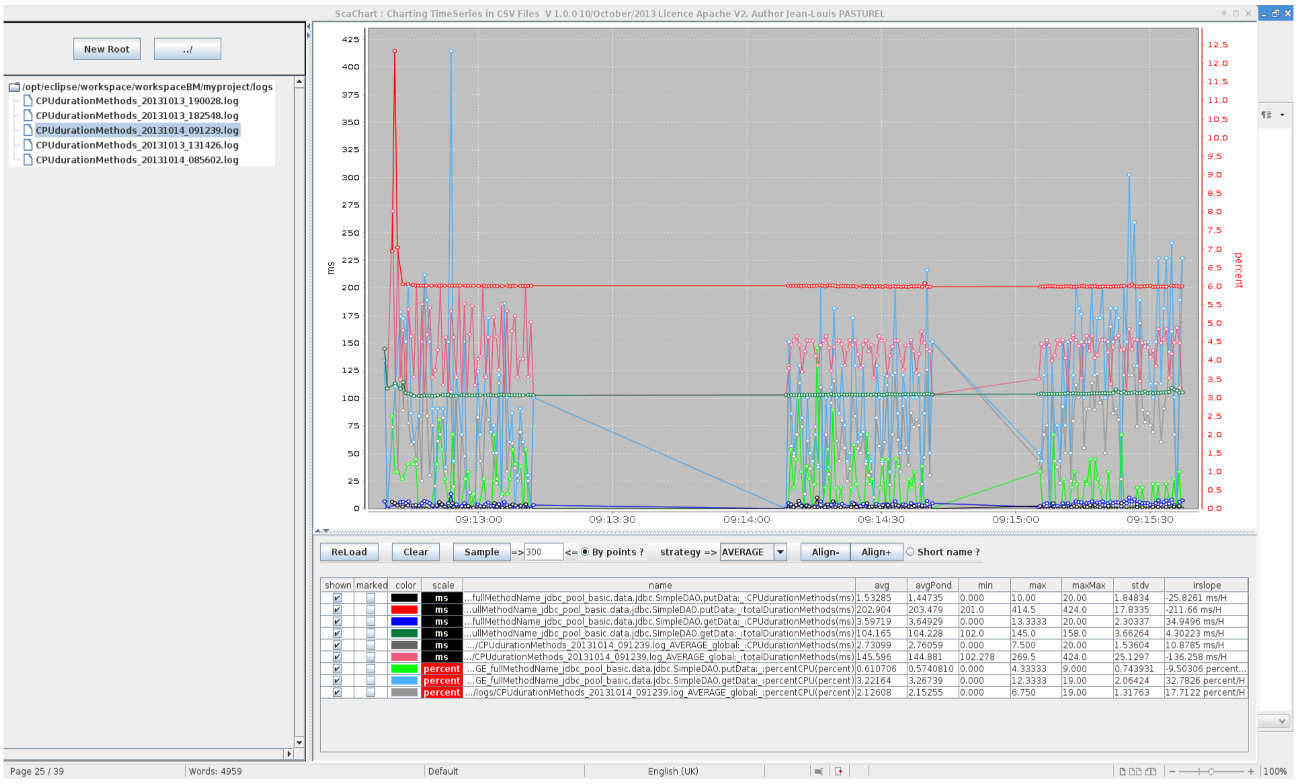
Csv file restrictions :
The only CSV files that Viewer and Dyn Viewer can graph, must have the following configuration :

- the first line contains title of columns separated by the csv separator
- the first column must be a date respecting one format described in the file **<packagePkg_Home>/config/scaChartDates.properties**
- The others columns can contains numeric values or strings (Pivots), but a column must be fill with the same type. This column can be empty (if ; is the csv separator ;; indicates an empty column)

This kind of file contains "**Time Series**", in sense of the framework JFreeChart concepts.

This 2 sub-menus are similar, the difference is that the Dyn Viewer sub-menu refreshes periodically the graph if the csv file has been updated. The period of refreshing is given by the variable : `scaviewer.dyn.timeout` in the file **<packagePkg_Home>/config/scaChart.properties**

DnD means Drag and Drop.
This menu permits to choose csv files, the tree at the left is positioned at the root folder of the current project.
Only the files, where the suffix is in the list of the variable `jtree.suffixes` , are displayed (file **<packagePkg_Home>/config/scaChart.properties)**


ScaChart : Charting TimeSeries in CSV Files V 1.0.0 10/October/2013 Licence Apache V2. Author Jean-Louis PASTUREL

/opt/eclipse/workspace/workspaceBM/myproject/logs
- CPUdurationMethods_20131013_190028.log
- CPUdurationMethods_20131013_182548.log
- CPUdurationMethods_20131014_091239.log
- CPUdurationMethods_20131013_131426.log
- CPUdurationMethods_20131014_085602.log

shown	marked	color	scale	name	avg	avgPond	min	max	maxMax	stdv	irslope
✓	☐	⬛	ms	...fullMethodName_jdbc_pool_basic.data.jdbc.SimpleDAO.putData:_:CPUdurationMethods(ms)	1.53285	1.44735	0.000	10.00	20.00	1.84834	-25.8261 ms/H
✓	☐	🟥	ms	...ullMethodName_jdbc_pool_basic.data.jdbc.SimpleDAO.putData:_:totalDurationMethods(ms)	202.904	203.479	201.0	414.5	424.0	17.8335	-211.66 ms/H
✓	☐	🟦	ms	...fullMethodName_jdbc_pool_basic.data.jdbc.SimpleDAO.getData:_:CPUdurationMethods(ms)	3.59719	3.64929	0.000	13.3333	20.00	2.30337	34.9496 ms/H
✓	☐	🟩	ms	...ullMethodName_jdbc_pool_basic.data.jdbc.SimpleDAO.getData:_:totalDurationMethods(ms)	104.165	104.228	102.0	145.0	158.0	3.66264	4.30223 ms/H
✓	☐	⬛	ms	.../CPUdurationMethods_20131014_091239.log_AVERAGE_global:_:CPUdurationMethods(ms)	2.73099	2.76059	0.000	7.500	20.00	1.53604	10.8785 ms/H
✓	☐	🟥	ms	.../CPUdurationMethods_20131014_091239.log_AVERAGE_global:_:totalDurationMethods(ms)	145.596	144.881	102.278	269.5	424.0	25.1297	-136.258 ms/H
✓	☐	🟩	percent	...GE_fullMethodName_jdbc_pool_basic.data.jdbc.SimpleDAO.putData:_:percentCPU(percent)	0.610706	0.5740810	0.000	4.33333	9.000	0.743931	-9.50306 percent...
✓	☐	🟦	percent	...GE_fullMethodName_jdbc_pool_basic.data.jdbc.SimpleDAO.getData:_:percentCPU(percent)	3.22164	3.26739	0.000	12.3333	19.00	2.06424	32.7826 percent/H
✓	☐	⬜	percent	.../logs/CPUdurationMethods_20131014_091239.log_AVERAGE_global:_:percentCPU(percent)	2.12608	2.15255	0.000	6.750	19.00	1.31763	17.7122 percent/H

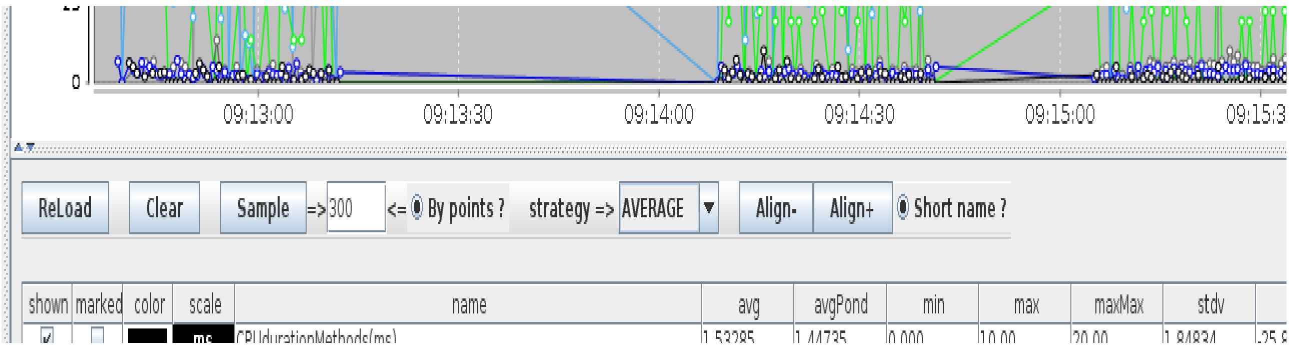
Page 25 / 39 Words: 4959 Default English (UK) 100%

After deploying a branch of a tree by clicking on node, you can drag and drop a csv file or a discontinued list of csv files to the right side of the screen. You can also add csv file by DnD from the same or another directory.

The graph must be zoomed by selecting a region. It can be de-zoomed by dragging on it from the right to the left.

Clicking right on the graph provides also other feature (copy in the clipboard ..)
Clicking right on the title of the table permits to add/remove columns.
Clicking right in the contains of the table permits remove selected row series on the table and on the graph.

Buttons :



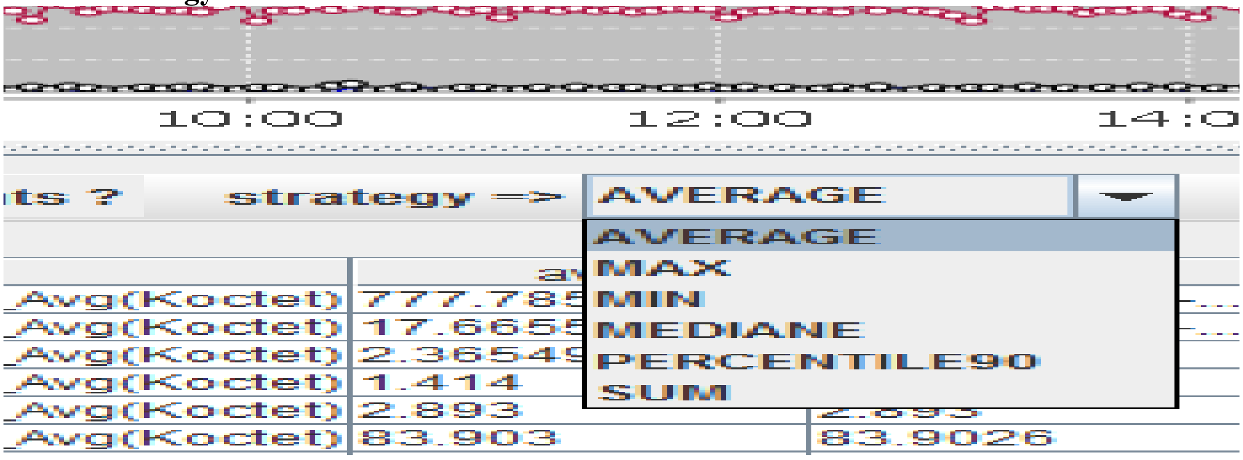
Reload
gives the possibility to reload the initial csv files
Clear
Cleanes all CSV file from the Chart and the table
Sample
The button is bound with the text field and the radio button at its right

- if the radio button “By nb max points ?” is selected, the chart is shown with the number of points given by the text field (in fact the time interval shown is divided in 300 intervals for the example above)
- if the radio button “By nb max points ?” is unselected, the time interval shown is divided in periods of the given by the text field expressed in milliseconds.

Hint :

When you zoom a part of the chart, click on **Sample** to get more points.

Strategy



When the csv files have more lines than can be displayed on chart (or the number in the text field described just above), you can choose the strategy of aggregation in this combo-box.

Align- :

allows, when 2 or more series are “marked” in the table (column “marked”), to align these series to the left side of the chart by shifting the beginning of all series to the minimum of the beginning of the marked series. It helps for comparison or correlation between series. It can be combined with the “Translate” feature of the table (§ 3.4.3).

Align+ :

allows, when 2 or more series are “marked” in the table (column “marked”), to align these series to the right side of the chart by shifting the beginning of all series to the maximum of the beginning of the marked series. It helps for comparison or correlation between series. It can be combined with the “Translate” feature of the table (§ 3.4.3).

Short Name:

allows, when clicked to show short name of the series, in the table.

3.5 Instrumentation of the target JVM

3.5.1 For all JVMs / Servers

You have two possible kinds of instrumentation with **byteman** :
- at startup of the JVM/Server with the JVM parameter **-javaagent**
- more dynamically, after the statrtup of the JVM / Server, with specific **byteman** script :
bminstall / bmjava / bmsubmit, cf Byteman documentation.

If the agent **mybyteman.jar** is installed in the directory **/tmp/**, the java VM arguments must contains at least :
-javaagent:/tmp/mybyteman.jar=resourcescript:jlp/byteman/helper/bytemanpkg.btm

If the script is out of the packaging mybyteman.jar (in /tmp/ for example), the javaagent could be :
-javaagent:/tmp/mybyteman.jar=script:/tmp/bytemanpkg.btm

In some cases, when application classes are needed (for Mock Object for example), you must add package for the classpath of the javaagent like :
-
javaagent:/tmp/mybyteman.jar=resourcescript:jlp/byteman/helper/bytemanpkg.btm,sys:/tmp /jdbc_pool_basic.jar

See Mock example, further in the document.

3.5.2 JBOSS 7.2+ / JBOSS-EAP 6.1+

You can set javaagent in standalone.conf like described just above, or you can use de dynamic submision (tab Remote Actions of the tool BytemanPkg).
For JBOSS 7.2+/ JBOSS-EAP 6.1+, needed by the architecture of JBOSS modules on the micro kernel, you have to add the JVM parameter :
-Djboss.modules.system.pkgs="org.jboss.byteman,jlp"
This parameter is set in file **<JBOSS_HOME>/bin/standalone.conf** or **<JBOSS_HOME>/bin/standalone.conf.bat**

3.5.3 JOnAS 5.1+

Needed by Osgi architecture, two files must be modified in **<JONAS_BASE>/conf/osgi**
defaults.properties :
complete the parameter **bootdelegation-packages** by :
org.apache.xpath.jaxp.*, \
**jlp, **
**jlp.*, **
**org.jboss.byteman, **
org.jboss.byteman.*

gateway.properties :
add the line below at the end of the file:
org.osgi.framework.bundle.parent app
You can set **javaagent** in bin/jonas at start level pragraph like described above, or you can use de dynamic submision (tab **Remote Actions** of the tool **BytemanPkg**).

3.5.4 TOMCAT 6/7

To include the javagent at startup, modify the start part of the script bin/catalina.sh as shown below.

```
elif [ "$1" = "start" ] ; then
        # Adding byteman agent
        JAVA_OPTS=$JAVA_OPTS"
-javaagent:/tmp/mybyteman.jar=resourcescript:jlp/byteman/helper/bytemanpkg.btm"
  if [ ! -z "$CATALINA_PID" ]; then
    if [ -f "$CATALINA_PID" ]; then
      if [ -s "$CATALINA_PID" ]; then
        echo "Existing PID file found during start."
```

Tomcat runs correctly also with the dynamic submission (without adding javaagent in catalina.sh) if the JVM is OpenJDK or Oracle JVM.
For IBM JVM see further for the issue.

3.5.5 WebSphere 8.5+

Quick way :

Manually modifying the file **server.xml** located in :

<WEBSPHERE_ROOT>/AppServer/profiles/<name AppSrv>/config/cells/<name Cell>/nodes/<name Node>/servers/<name server>/server.xml

Retrieve the attribute **genericJvmArguments and add the parameters below :**

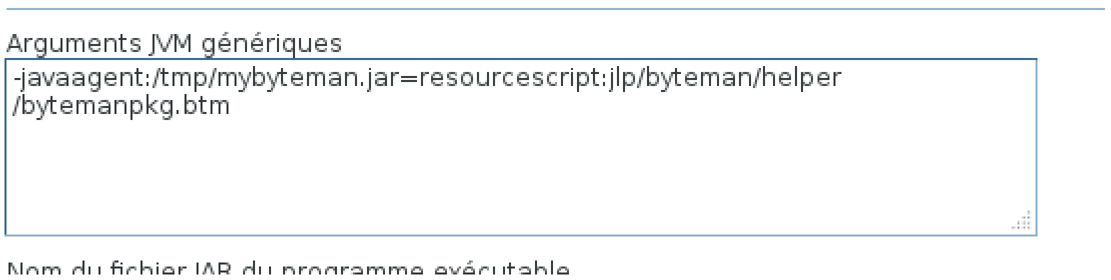
```
<process:Server>
...
        <processDefinitions>
        ...
                <jvmEntries ... genericJvmArguments=”...”>
```

```
genericJvmArguments="-
javaagent:/tmp/mybyteman.jar=resourcescript:jlp/byteman/helper/bytemanpkg.btm"
```

Be care to respect the XML structure of this file

Normal way by the admin console :

The chainned menus are :
On the left Panel :
servers ->servers types -> Websphere Application servers.
Choose your server in the right panel. And after in the right panel :
Process gestion and Java -> Process definition -> Java Virtual Machine ->
In the text field : Generic JVM arguments, fill as shown below :


Arguments JVM génériques

```
-javaagent:/tmp/mybyteman.jar=resourcescript:jlp/byteman/helper
/bytemanpkg.btm
```

Nom du fichier JAR du programme exécutable

Nota: My WebSphere installation is with French Language, so the labels must be a bit different in English

Due to a JVM IBM brhaviour, the dynamic submission doesn't run with Websphere that uses an embedded IBM JVM for OS : Windows, Linux, AIX. It could run on SOLARIS, or HP-UNIX but not tested.

3.5.6 WebLogic 12c

Quick way if the Application server is started from script startWebLogic.sh :

Manually modifying the file **startWebLogic.sh** located in :

<DOMAINS_ROOT>/<My_Domain>/bin/startWebLogic.sh

Modifying after the comment # START WEBLOGIC

```
# START WEBLOGIC


echo "starting weblogic with Java version:"


${JAVA_HOME}/bin/java ${JAVA_VM} -version
BYTEMAN_OPTS="-
javaagent:/tmp/mybyteman.jar=resourcescript:jlp/byteman/helper/bytemanpkg.btm "
if [ "${WLS_REDIRECT_LOG}" = "" ] ; then
        echo "Starting WLS with line:"
        echo "${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} ${BYTEMAN_OPTS}
```

```
-Dweblogic.Name=${SERVER_NAME} -Djava.security.policy=${WLS_POLICY_FILE} ${JAVA_OPTIONS} $
{PROXY_SETTINGS} ${SERVER_CLASS}"
        ${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS}  ${BYTEMAN_OPTS} -Dweblogic.Name=$
{SERVER_NAME} -Djava.security.policy=${WLS_POLICY_FILE} ${JAVA_OPTIONS} ${PROXY_SETTINGS}
${SERVER_CLASS}
else
        echo "Redirecting output from WLS window to ${WLS_REDIRECT_LOG}"
        ${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS}  ${BYTEMAN_OPTS}
-Dweblogic.Name=${SERVER_NAME} -Djava.security.policy=${WLS_POLICY_FILE} ${JAVA_OPTIONS} $
{PROXY_SETTINGS} ${SERVER_CLASS}  >"${WLS_REDIRECT_LOG}" 2>&1
fi
```

Other way when there is an Admin server and at least an Application server (cluster) by the admin console :

The chainned menus are :
On the left Panel :
Environments -> servers ->.
Choose your Application server in the right panel. And after in the right panel :
Upper Tab Configuration -> Sub-Tab Start Serverd
In the text field : Arguments Set :

```
-javaagent:/tmp/mybyteman.jar=resourcescript:jlp/byteman/helper/bytemanpkg.btm
```

Re-start the Application server from the console.
I have not tested, because i have installed a standalone server (containing both Admin + Application)

Nota : Weblogic 12c also supports the dynamic submission of Rules.

3.5.7 GlassFish 4.0 JEE7 (needs JDK 1.7 at least)

Normal Way => Admin console:

http://localhost:4848

In Left Panel :
Configurations-> Server-config -> JVM-Settings
 Right Panel :
Choose Tab JVM Options -> Add JVM Option
Fill the new Option with:

```
-javaagent:/tmp/mybyteman.jar=resourcescript:jlp/byteman/helper/bytemanpkg.btm
```

Glassfish is built around an OSGi architecture, Byteman classes and Helper Classes must be loaded by the system classloader.

There are 2 Solutions :

Adding another JVM Option :

```
-Xbootclasspath/a:/tmp/mybyteman.jar
```

mybyteman.jar contains all classes of byteman.jar (packages org.jboss.byteman*) and my custom Helpers (packages jlp.*)

Modifying configuration osgi.properties :

There are 2 modifications (**in red color**) to do in file :
<GLASSFISH_ROOT>/glassfish/config/osgi.properties

```
# There is no need to use bootdelegation except for the following issues:
# 1. EclipseLink
# 4. NetBeans profiler packages exist in parent class loader (see issue #8612)
# 5. BTrace exists in bootclasspath.
org.osgi.framework.bootdelegation=${eclipselink.bootdelegation}, \
                    com.sun.btrace, com.sun.btrace.*, \
                    org.netbeans.lib.profiler,
        org.netbeans.lib.profiler.*, \
                                        org.jboss.byteman, org.jboss.byteman.*, \
                                        jlp, jlp.*


# The OSGi R4.2 spec says boot delegation uses the boot class loader by default. We need
# to configure it to use the framework class loader because that class loader is
# configured with extra classes like jdk tools.jar, derby jars, etc. that must be
# made available in GlassFish to work.
# framework is the original value.
# org.osgi.framework.bundle.parent=framework
org.osgi.framework.bundle.parent=app
```

The two solutions run for me for the javaagent.

Quick configuration :

The console operation modify the XML file :
<GLASSFISH_ROOT>glassfish/domains/<domain_name>/config/domain.xml
The JvmOptions for the server are located at :

```
<configs>
```

```
<config name="server-config">
    <java-config ...>
        <jvm-options> ...
```

So you can add **jvm-options** as for exmaple

```
<jvm-options>-javaagent:/tmp/mybyteman.jar=resourcescript:jlp/byteman/helper/bytemanpkg.btm</jvm-options>
<jvm-options>-Xbootclasspath/a:/tmp/mybyteman.jar</jvm-options>
```

You need to re-start the GlassFish domain.

Nota : for dynamic submission, you must :
- add the JVM option

```
    -Xbootclasspath/a:/tmp/mybyteman.jar
```

- do the 2 modifications in file **<GLASSFISH_ROOT>/glassfish/config/osgi.properties**

3.5.8 Eclipse/JETTY 9.1

To include the javagent at startup, modify the part of the file **$JETTY_BASE/start.ini** as shown below.

```
--exec
-Xms128M
-Xmx128M
-Djetty.base=/opt/jetty9/base
-javaagent:/tmp/mybyteman.jar=resourcescript:jlp/byteman/helper/bytemanpkg.btm
```

Jetty runs correctly also with the dynamic submission (without adding javaagent in **start.ini**) if the JVM is OpenJDK or Oracle JVM.
The command to get the Jetty PID is

```
ps -feww | grep java | grep org.eclipse.jetty.xml.XmlConfig | tr -s ' ' | cut -d ' ' -f2
```

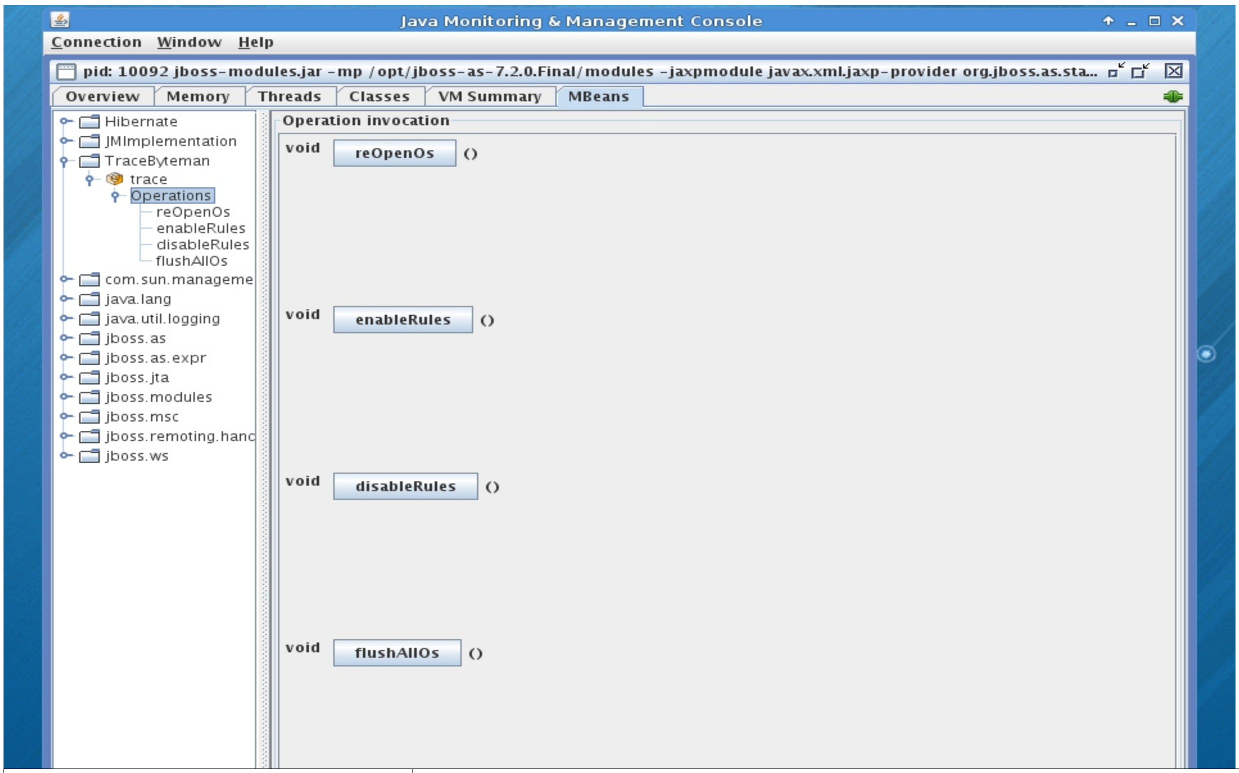
For IBM JVM see further for the issue.

3.6 Monitoring rules execution

The custom Helper MyHelper uses a class Trace which is a Mbean.

3.6.1 Monitoring throw JConsole

This Mbean exposes 4 operations as shown in the JConsole below :



Operations	Comment
reOpenOs	means re-open OutputStream. This operation flushes and closes all current outputstream of the trace files and re-open new ones. Useful, for example to start tracing at certain point, eliminating the trace of the starting of the server. If trace are in **gzip mode**, this operation is **necessary** to get a correct gzip file.
enablesRules	All rule script of the bytemanPkg have in their IF clause the test : IF activeRules AND ... this operation set activeRules to true. The DO statement of the rule is excecuted
disableRules	All rule script of the bytemanPkg have in their IF clause the test : IF activeRules AND ... this operation set activeRules to false. The Do statement of the rule is not excecuted.
flushAllOs	means flushes OutputStream. This operation flushes all current outputstream of the trace files. Useful when **gzip mode is not checked**, to display current trace files.

3.6.2 Monitoring throw script and AttachAPI

We can access to the Mbean using a script , calling the Mbean throw the Attach API (included in tools.jar of the JDK). So a JDK is necessary on the remote servers.

The script is (for Linux) :
attachMBean.sh
It must be launched, on the remote servers, with the same operating system user than the java process to attach.
The command is :

```
attachMBean.sh <PID_JAVA> <OPERATION>

or

attachMBean.sh  <OPERATION>
```

in last case, it attaches the first java process with a javaagent / mybyteman.jar in the JVM parameters.
The four operations described just above are available.

These features are also integrated , in the Tab Remote Actions of the Gui of BytemanPkg.

4 ANNEXES

4.1 Some more explained configuration rules.

4.1.1 Common informations for all rules

Definition of a trigger.

Almost of Dialog for configuring rules have a TextArea or a TextField named listMethods.
When listMethods is a TextArea, you can fill several methods, separated by semicolon (and a Line Feed if you want).

The complete pattern of a method, for bytemanPkg is :
opt means optionnal.

opt(<return-Type>) opt(^) (<full_path_Class>|<shortName_Class>).<method> opt((Type_arg1,Type_arg2,...)) ;

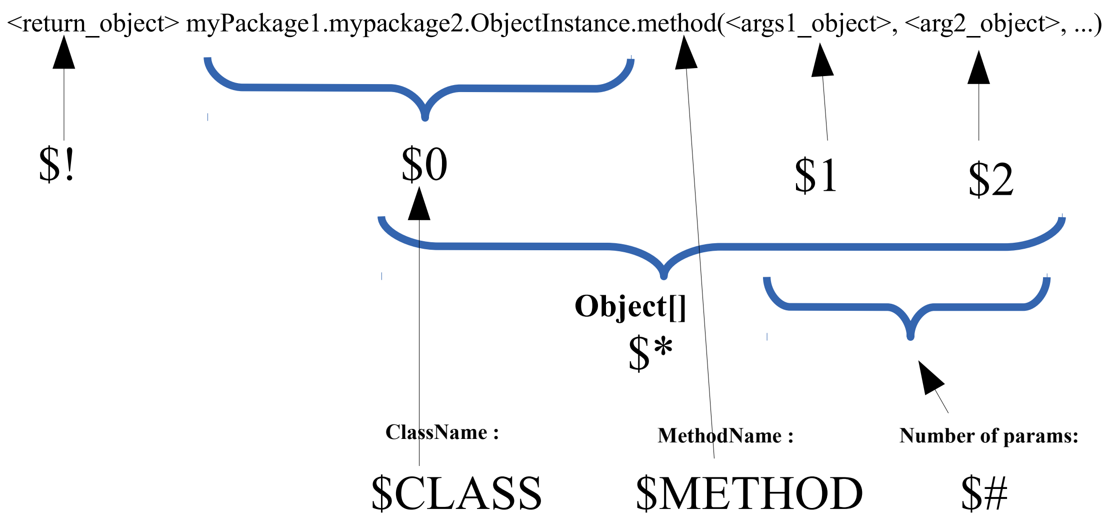
Examples:
a full definition
int ^mypackage1.mypackag2.MyClass1.myMethod1(String,int);

trigger the method myMethod1, with 2 parameters String,int , belonging to all classes extending or implementing the class/interface mypackage1.mypackage2.MyClass1 and returning an int.

A more simple definition
MyClass1.myMethod2;
trigger all the methods myMethod1, belonging to the class MyClass1, and belonging to any package.

Definition of some byteman variables.

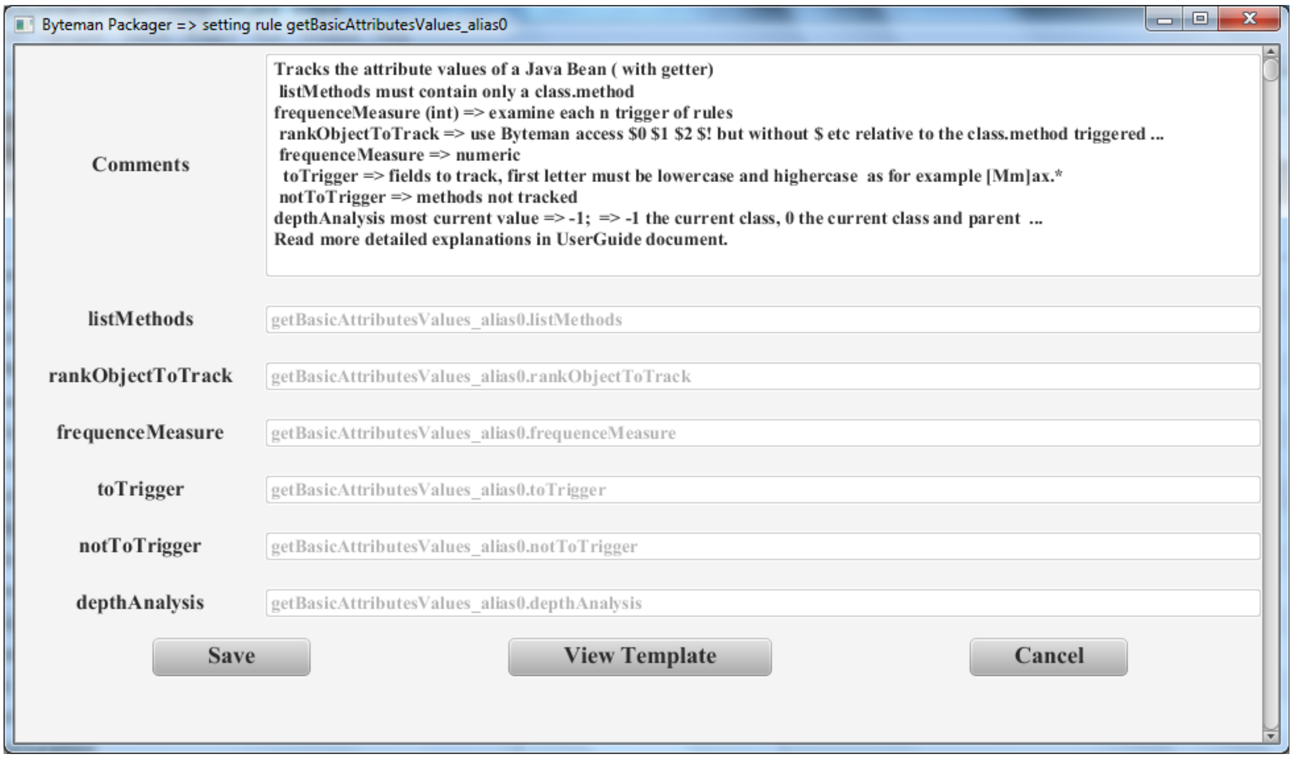
For more information read the document =>
http://downloads.jboss.org/byteman/2.1.3/ProgrammersGuide-2.1.3.1.pdf



<return_object> myPackage1.mypackage2.ObjectInstance.method(<args1_object>, <arg2_object>, ...)
$!
$0
$1
$2
Object[]
$*
ClassName :
$CLASS
MethodName :
$METHOD
Number of params:
$#

4.1.2 Rule GetBasicAttributes

Below, the Dialog to fill :



Byteman Packager => setting rule getBasicAttributesValues_alias0

Comments

Tracks the attribute values of a Java Bean (with getter)
listMethods must contain only a class.method
frequenceMeasure (int) => examine each n trigger of rules
rankObjectToTrack => use Byteman access $0 $1 $2 $! but without $ etc relative to the class.method triggered ...
frequenceMeasure => numeric
toTrigger => fields to track, first letter must be lowercase and highercase as for example [Mm]ax.*
notToTrigger => methods not tracked
depthAnalysis most current value => -1; => -1 the current class, 0 the current class and parent ...
Read more detailed explanations in UserGuide document.

listMethods	getBasicAttributesValues_alias0.listMethods
rankObjectToTrack	getBasicAttributesValues_alias0.rankObjectToTrack
frequenceMeasure	getBasicAttributesValues_alias0.frequenceMeasure
toTrigger	getBasicAttributesValues_alias0.toTrigger
notToTrigger	getBasicAttributesValues_alias0.notToTrigger
depthAnalysis	getBasicAttributesValues_alias0.depthAnalysis

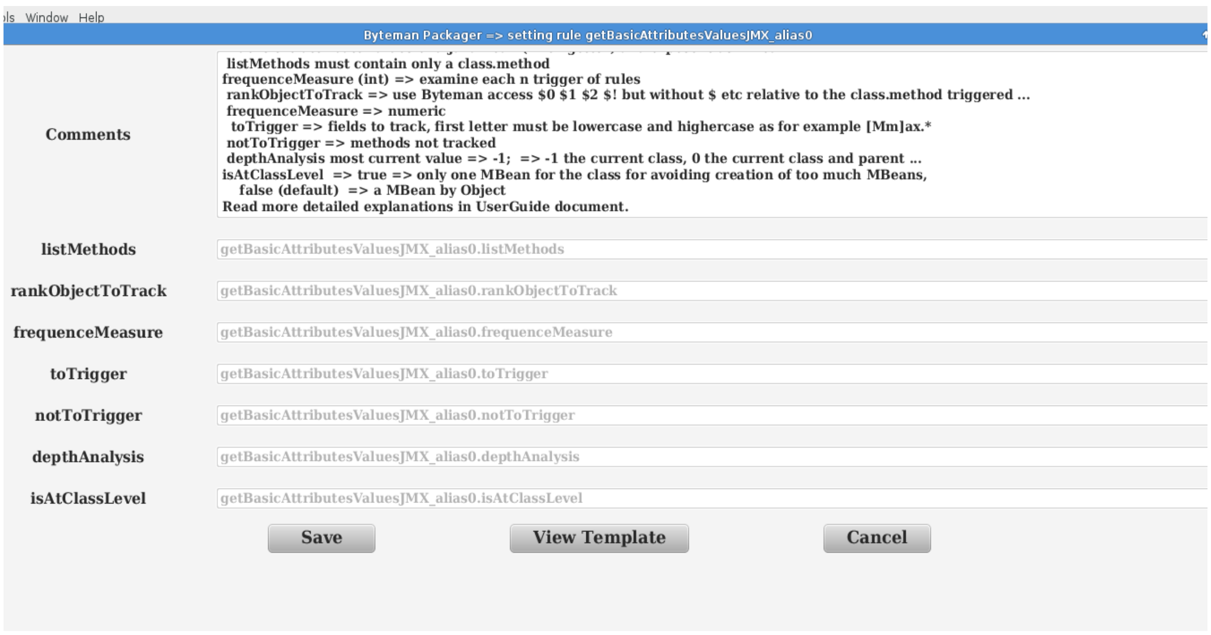
This rules is used to display values of basic attributes of an object. Basic attributes means :
- Number => int,long,double,float and their Class Wrapper,

- String
- boolean
- if the attribute is not a basic type, the method toString is applied.

Field	Comment
listMethods	In this rule, only a method is allowed. This method must be chosen carefully, because the attributes tracked may be : - those of the running object => $0 - those of one of the parameters of the method ($1, $2 ...)
rankObjectToTrack	Integer, and as said above , 0 or 1 or 2 etc. (for $0,$1,$2 ..)
frequenceMeasure	Integer, the frequency of measure / logging when the rule is triggered.
toTrigger	List of pattern of attributes to track, separated by semicolon. Pattern are java Pattern
notToTrigger	List of java Pattern to excludes. Useful when the pattern of the method triggered could be triggered (infinite recursion), or when others attributes are not useful
depthOfAnalysis	Integer, the general case you set it to -1 (the fields and the getter are in the currentt class), 0 means that the getters are searched in the current class and it parent class , 1 => current/parent/parent of parent etc..

4.1.3 Rule GetBasicAttributesJMX

Belolw, the Dialog to fill :


ls Window Help
Byteman Packager => setting rule getBasicAttributesValuesJMX_alias0
Save
View Template
Cancel

Comments	listMethods must contain only a class.method frequenceMeasure (int) => examine each n trigger of rules rankObjectToTrack => use Byteman access $0 $1 $2 $! but without $ etc relative to the class.method triggered ... frequenceMeasure => numeric toTrigger => fields to track, first letter must be lowercase and highercase as for example [Mm]ax.* notToTrigger => methods not tracked depthAnalysis most current value => -1; => -1 the current class, 0 the current class and parent ... isAtClassLevel => true => only one MBean for the class for avoiding creation of too much MBeans, false (default) => a MBean by Object Read more detailed explanations in UserGuide document.
listMethods	getBasicAttributesValuesJMX_alias0.listMethods
rankObjectToTrack	getBasicAttributesValuesJMX_alias0.rankObjectToTrack
frequenceMeasure	getBasicAttributesValuesJMX_alias0.frequenceMeasure
toTrigger	getBasicAttributesValuesJMX_alias0.toTrigger
notToTrigger	getBasicAttributesValuesJMX_alias0.notToTrigger
depthAnalysis	getBasicAttributesValuesJMX_alias0.depthAnalysis
isAtClassLevel	getBasicAttributesValuesJMX_alias0.isAtClassLevel

This rules is used to expose a Mbean that contains the values of basic attributes of an object. Basic attributes means :
- Number => int,long,double,float and their Class Wrapper,
- String
- boolean
- if the attribute is not a basic type, the method toString is applied.
The Mbean is available in a JConsole, or by requesting the system MbeanServer with JMX.

The first 6 parameters are the the same as in rule GetBasicAttributeJMX.
There is a seventh parameter that configure the Mbean output.

Field	Comment
listMethods	In this rule, only a method is allowed. This method must be chosen carefully, because the attributes tracked may be : - those of the running object => $0 - those of one of the parameters of the method ($1, $2 ...)
rankObjectToTrack	Integer, and as said above , 0 or 1 or 2 etc. (for $0,$1,$2 ..)
frequenceMeasure	Integer, the frequency of measure / logging when the rule is triggered.
toTrigger	List of pattern of attributes to track, separated by semicolon. Pattern are java Pattern
notToTrigger	List of java Pattern to excludes. Useful when the pattern of the method triggered could be triggered (infinite recursion), or when others attributes are not useful
depthOfAnalysis	Integer, the general case you set it to -1 (the fields and the getter are in the currentt class), 0 means that the getters are searched in the current class and it parent class , 1 => current/parent/parent of parent etc..
IsAtClassLevel	Boolean : true => only one MBean for the class for avoiding creation of too much Mbeans => false (default) => a MBean by Object but limited to 100 Objects; After 100 Objects, the Mbean are destroyed, and new Mbeans are created staring with 0

4.1.4 Rules `mockMethods / mockMethodsWithTrace`

The two templates are in fact like dummy templates, all the sections of the rule must be filled.

Mock a methods is useful to simulate interfaces with external application during test.
The simplest method to mock, are the methods that returns void, or a standard Object of the Java

Standard API.
I give an example of mocking a method, with a specific Helper, that returns a collection of String.I modify the return of the method, that does not read the datas in the database : the datas are build by the helper.
 The method to mock is :

```
public Collection getData()
                throws Exception
        {
                nbGet++;
                // generate a Random wait 0-1s
                try {
                Thread.sleep(100L + (long) Math.random() * 100);
                }
                catch(InterruptedException e)
                {}
                System.out.println("nbGet="+nbGet);
                UselessClass useless=new UselessClass(nbGet);
                useless.doSomeThing();
                Connection conn = null ;
                Statement sql = null ;

                try{

                        conn = _ds.getConnection() ;
                        sql = conn.createStatement() ;
                        ResultSet rs = sql.executeQuery( _sql.get( "fetch_data" ) ) ;

                        ArrayList result = new ArrayList() ;

                        while( rs.next() ){

                                result.add(
                                        new SimpleDTO(
                                                rs.getString( "names" ) ,
                                                rs.getInt( "numbers" )
                                        )
                                        ) ;

                        }

                        return( result ) ;

                }catch( Throwable t ){

                        throw( new Exception( t ) ) ;

                }finally{

                        DbUtil.close( sql ) ;
                        DbUtil.close( conn ) ;

                }

        }
```

The helper used to mock the method :

```
package jlp.byteman.helper.specific;

import java.lang.reflect.Constructor;
import java.lang.reflect.InvocationTargetException;
import java.util.ArrayList;


// no  import a specific applicative Object.
```

```
import org.jboss.byteman.rule.Rule;

import jlp.byteman.helper.MyHelper;

public class MockExampleReflect extends MyHelper {

	protected MockExampleReflect(Rule rule) {
		super(rule);


	}

	public java.util.Collection getMock1Collection()
	{
		ArrayList arrL=new ArrayList();
		// Need of a specific applicative Object.
		// The jar of the object must be in the classpath to compile
		// At runtime, on the target server you may add sys:<pathToJarcontainingtheObject> in the
javaagent parameters

		Class<?> clazz = null;;
		try {
			clazz =
Class.forName("jdbc_pool_basic.core.SimpleDTO",false,ClassLoader.getSystemClassLoader());
			Constructor ctor=clazz.getConstructor(String.class,Integer.TYPE);
			for(int i=0;i<10;i++){
			arrL.add(ctor.newInstance("Mock-Reflection-"+i,i));
			}
		} catch (ClassNotFoundException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (NoSuchMethodException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (SecurityException e) {
			// TODO Auto-generated catch block
```

```
                        e.printStackTrace();
                } catch (InstantiationException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (IllegalAccessException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (IllegalArgumentException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (InvocationTargetException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }


                return arrL;
        }
}
```

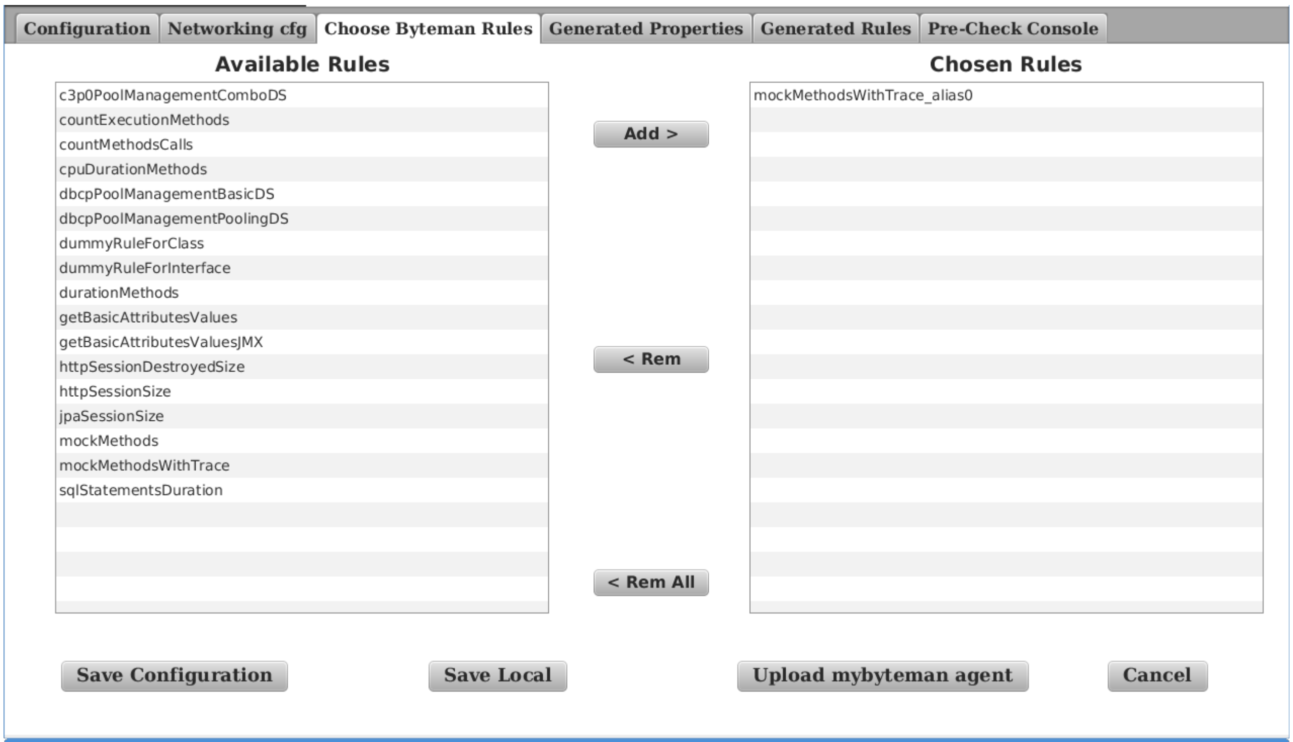
SimpleDTO :

```
public class SimpleDTO {

private final String _name ;
private final int _number ;
private final String _stringRep ;

public SimpleDTO( final String name , final int number ){

        _name = name ;
        _number = number ;
        _stringRep = "[SimpleDTO: " + _name + "/" + _number + "]" ;

        return ;

} // ctor
public String getName(){
        return( _name ) ;
} // getName()

public int getNumber(){
        return( _number ) ;
} // getNumber()

public String toString(){
        return( _stringRep ) ;
} // toString()

} // public class SimpleDTO
```

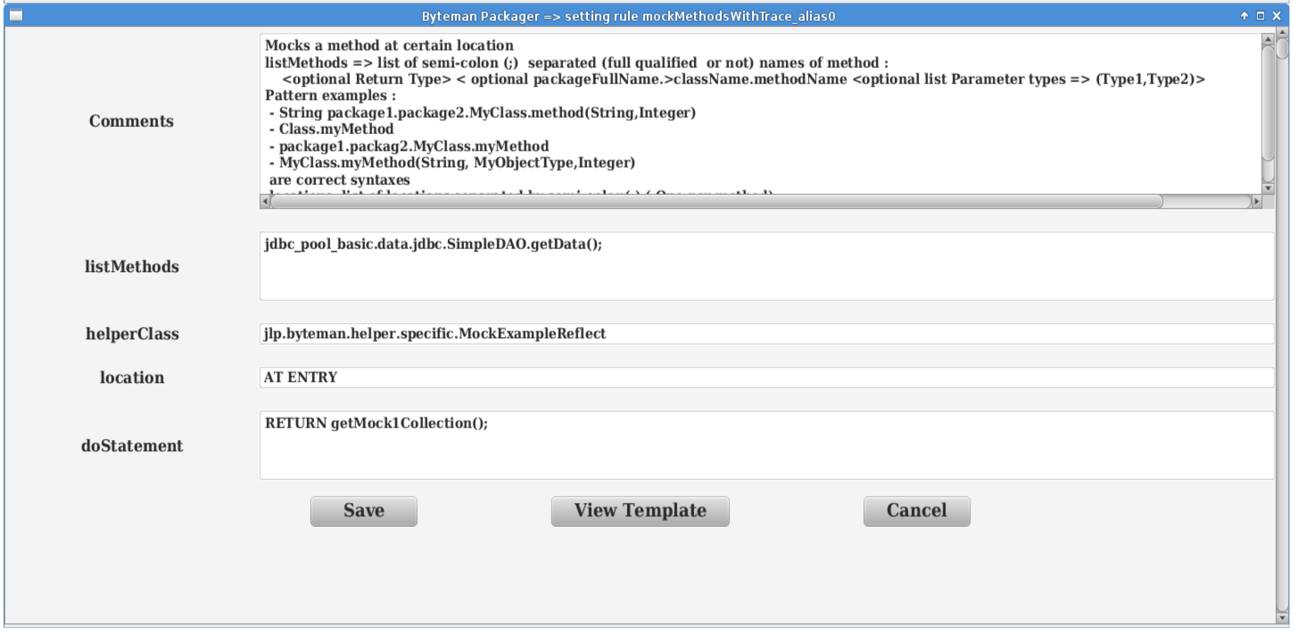
In libExt folder, there are two archives used fot this example :
- jdbc_pool_basic.war (the war application)
- jdbc_pool_basic.jar (containts SimpleDTO)

To get an Web-App example, I started with this example :
https://today.java.net/pub/a/today/2005/11/17/app-managed-datasources-with-commons-dbcp.html#resources

Building the rule :



Available Rules	Chosen Rules
c3p0PoolManagementComboDS	mockMethodsWithTrace_alias0
countExecutionMethods	
countMethodsCalls	
cpuDurationMethods	
dbcpPoolManagementBasicDS	
dbcpPoolManagementPoolingDS	
dummyRuleForClass	
dummyRuleForInterface	
durationMethods	
getBasicAttributesValues	
getBasicAttributesValuesJMX	
httpSessionDestroyedSize	
httpSessionSize	
jpaSessionSize	
mockMethods	
mockMethodsWithTrace	
sqlStatementsDuration	



Byteman Packager => setting rule mockMethodsWithTrace_alias0

Comments	Mocks a method at certain location listMethods => list of semi-colon (;) separated (full qualified or not) names of method : <optional Return Type> < optional packageFullName.>className.methodName <optional list Parameter types => (Type1,Type2)> Pattern examples : - String package1.package2.MyClass.method(String,Integer) - Class.myMethod - package1.packag2.MyClass.myMethod - MyClass.myMethod(String, MyObjectType,Integer) are correct syntaxes
listMethods	jdbc_pool_basic.data.jdbc.SimpleDAO.getData();
helperClass	jlp.byteman.helper.specific.MockExampleReflect
location	AT ENTRY
doStatement	RETURN getMock1Collection();

Save / Save Configuration / Upload mybyteman agent.

The rule generated for the Mock Object is :

```
RULE mockMethodsWithTrace_1 jdbc_pool_basic.data.jdbc.SimpleDAO.getData() 2
```

```
        CLASS jdbc_pool_basic.data.jdbc.SimpleDAO
        METHOD getData()
        HELPER jlp.byteman.helper.specific.MockExampleReflect

        AT ENTRY
                BIND fullName:String=$CLASS+".getData()";
                vrai:boolean=createCounter(fullName);
                newValue:int=incrementCounter(fullName);
                activeRules:boolean=getTrace().activeRules;
                sep:String=getProps().getProperty("bytemanpkg.csvSep", ";");
                title:String="date"+sep+"fullMethodName"+sep+"nb of mocking (unit)"+sep;
        IF activeRules
                DO
                        getTrace().openTrace(title,"mockMethodsWithTrace.log");
                        getTrace().append(title,currentDate()
+sep+fullName+sep+Integer.toString(newValue)+sep);
                RETURN getMock1Collection();

ENDRULE
```

For my test, i use the WAS JBOSS-as 7.2 community. I put the archive **jdbc_pool_basic.war** in the **deployments** directory.
In standalone.conf the agent is parametred as bellow :

```
 JBOSS_MODULES_SYSTEM_PKGS="org.jboss.byteman,jlp"
BYTEMAN_OPTS="
-javaagent:/tmp/mybyteman.jar=script:jlp/byteman/helper/bytemanpkg.btm,sys:/opt/ecli
pse/workspace/TestDBCPBasic/dist/jdbc_pool_basic.jar
-Dorg.jboss.byteman.verbose -Dorg.jboss.byteman.debug “
JAVA_OPTS="$JAVA_OPTS
-Djboss.modules.system.pkgs=$JBOSS_MODULES_SYSTEM_PKGS $BYTEMAN_OPTS
-XX:+StartAttachListener  -Djava.awt.headless=true"
```

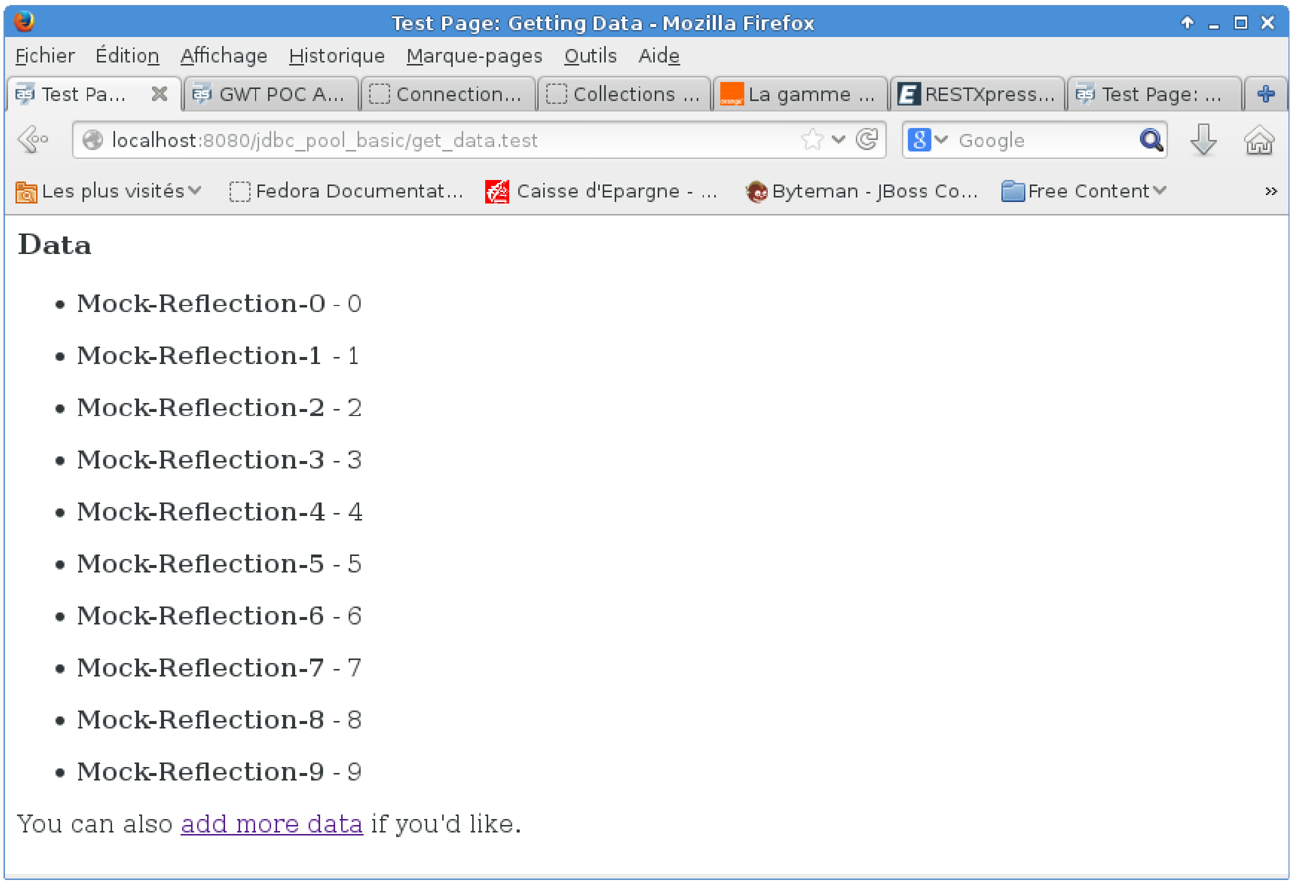
The jar (jdbc_pool_basic.jar) containing the class (SimpleDTO) used by the Mock Object must be in the classpath of the javaagent.

The verbose/debug mode is just for testing the rule, in heavy stressing, these parameters must be eliminated.

Launching JBOSS (nohup.out extracts) :

```
JAVA_OPTS:  -server -XX:+UseCompressedOops -Xms64m -Xmx512m -XX:MaxPermSize=256m
-Djava.net.preferIPv4Stack=true  -Djboss.modules.system.pkgs=org.jboss.byteman,jlp
-javaagent:/tmp/mybyteman.jar=script:jlp/byteman/helper/bytemanpkg.btm,sys:/opt/eclipse/workspace/TestD
BCPBasic/dist/jdbc_pool_basic.jar -Dorg.jboss.byteman.verbose -Dorg.jboss.byteman.debug   -XX:
+StartAttachListener  -Djava.awt.headless=true
...
#[0m#[0m17:30:24,498 INFO  [stdout] (MSC service thread 1-3) org.jboss.byteman.agent.Transformer :
possible trigger for rule mockMethodsWithTrace_1 jdbc_pool_basic.data.jdbc.SimpleDAO.getData() 3 in
class jdbc_pool_basic.data.jdbc.SimpleDAO
#[0m#[0m17:30:24,531 INFO  [stdout] (MSC service thread 1-3)
RuleTriggerMethodAdapter.injectTriggerPoint : inserting trigger into
jdbc_pool_basic.data.jdbc.SimpleDAO.getData() java.util.Collection for rule mockMethodsWithTrace_1
jdbc_pool_basic.data.jdbc.SimpleDAO.getData() 3
#[0m#[0m17:30:24,537 INFO  [stdout] (MSC service thread 1-3) org.jboss.byteman.agent.Transformer :
inserted trigger for mockMethodsWithTrace_1 jdbc_pool_basic.data.jdbc.SimpleDAO.getData() 3 in class
jdbc_pool_basic.data.jdbc.SimpleDAO
```

Connecting to the application/puDatat/ getData (getData is the mocked method). The last screen is :


Test Page: Getting Data - Mozilla Firefox
Fichier Édition Affichage Historique Marque-pages Outils Aide
Test Pa... GWT POC A... Connection... Collections ... La gamme ... RESTXpress... Test Page: ...
localhost:8080/jdbc_pool_basic/get_data.test
Google
Les plus visités Fedora Documentat... Caisse d'Epargne - ... Byteman - JBoss Co... Free Content

Data

- Mock-Reflection-0 - 0
- Mock-Reflection-1 - 1
- Mock-Reflection-2 - 2
- Mock-Reflection-3 - 3
- Mock-Reflection-4 - 4
- Mock-Reflection-5 - 5
- Mock-Reflection-6 - 6
- Mock-Reflection-7 - 7
- Mock-Reflection-8 - 8
- Mock-Reflection-9 - 9

You can also add more data if you'd like.

that corresponds to the data returned by the mock Object and not these put in the database.

4.2 Linux : Remote access with su / sudo with JSch

Using the tab “Remote Actions”, you have to connect with the user that runs the JVM that you want to instrument, or you want to access locally with the Java Attach API.
We use de Jsch API for the remote access and we have 6 cases to treat :

- case 0 : you know the user/password of the targetted JVM => no need to use su / sudo
- case 1 : you know the user of the target JVM, you don't know the password of this user, you know the root password and root is **allowed** for remote connections.
- Case 2 : the targeted user is root , root **is not allowed** for remote connexion, you know root password, you know a user login/password that is allowed to do su.
- Case 3 : the targeted user is root , root **is not allowed** for remote connexion, you don't know root password, you know a user login/password that is allowed to do sudo su.
- Case 4 : you know the user of the target JVM, you don't know the password of this user, you know the root password and root is **not allowed** for remote connections, and you know a login/password of a third user with su enabled
- Case 5 : you know the user of the target JVM, you don't know the password of this user, you know the root password and root is **not allowed** for remote connections, and you know a login/password of a third user with sudo su enabled

For the last 5 cases, there are configurations that must be set.

4.2.1 Case 1 : password of user unknown, root password known and root allowed for remote connexion

Case : root → user

To allow **root** for remote connexion you have to set in the file (**root** login mandatory) :
/etc/ssh/sshd_config

```
PermitRootLogin yes
```

The remote action with a Jsch command looks like **:**
"su - "+user+" -c \"<commands>\""

4.2.2 Case 2 : targeted user root, root password known and not allowed for remote connexion, second user login/password known with su enabled

Case : user(su) → root

The second user (user) must be able to do su:
The setting is in the file (**root** login mandatory) : **/etc/pam.d/su**
this line below must be activated

```
auth              requisite       pam_wheel.so root_only group=staff trust use_uid
```

This second **user must be** also in the **group staff**.

"su - -c \"<commands>\""
and you have to fill the **root password** at prompt

4.2.3 Case 3 : targeted user root, root password known and not allowed for remote connexion, second user login/password known with sudo su enabled

Case : user(sudo su) → root

The second user (user) must be able to do **sudo su**:
The setting is in the file : **/etc/sudoers**
To not destroy the file, you must open it with the command (root login mandatory) : visudo -f /etc/sudoers . Don't use vi directly.
In this example the user is **JLP2** and you have to add this line in the corresponding sections :

```
#User_Alias
User_Alias JLPS  = JLP2
...
#Cmnd_Alias
Cmnd_Alias SU = /bin/su
...
# Commands Section
JLPS    ALL=SU
```

The remote action with a Jsch command looks like **:**
“sudo -S -p ' ' su -p -c \”<command>\””

and you have to fill the **user password** at prompt

4.2.4 Case 4 : password of user unknown, root password known and not allowed for remote connexion, third user login/password known with su enabled

Case : user3(su) → root → user
The third user (user3) is a user allowed to do a su. The setting is in the file (**root** login mandatory) : **/etc/pam.d/su**
this line below must be activated

```
auth            requisite       pam_wheel.so root_only group=staff trust use_uid
```

This third user(user3 must be also in the group **staff.**
The remote action with a Jsch command looks like **:**

```
"su  -p -c " + "\'(su - "+<USER:user>+" -c "+ command+")\'"
```

and you have to fill the **root password** at prompt

4.2.5 Case 5: password of user unknown, root password known and not allowed for remote connexion, third user login/password known with sudo su enabled

Case : user3(sudo su) → root → user
The third user (user3) must be able to do **sudo su**:
The setting is in the file : **/etc/sudoers**
To not destroy the file, you must open it with the command (root login mandatory) : visudo -f /etc/sudoers . Don't use vi directly.
In this example the user is **JLP2** and you have to add this line in the corresponding sections :

```
#User_Alias
User_Alias JLPS  = JLP2
…
#Cmnd_Alias
Cmnd_Alias SU = /bin/su
…
# Commands Section
JLPS    ALL=SU
```

The remote action with a Jsch command looks like **:**

```
 "sudo -p ' ' -S su  -p -c " +"\'(su - "+<USER:user>+" -c "+ command+")\'"
```

and you have to fill the **user password** at prompt

4.3 Little number of realized tests

JVM	WAS	javaagent	Install/Submit
Oracle JDK 6/7	JBOSS 7.2 / JONAS 5.1+,TOMCAT 6/7/,WebLogic12c,GlassFish 4.0	**OK**	**OK**
IBM JDK 6/7	JBOSS 7.2; WebSphere 8.5.5	**OK**	**KO***
Open JDK 6/7	JBOSS 7.2 / JONAS 5.1+,TOMCAT 6/7,WebLogic12c,GlassFish 4.0	**OK**	**OK**

*I got an error :

```
 14:56:36,458 ERROR [stderr] (Attachment 51643)
java.lang.UnsupportedOperationException: cannot get the
capability, performing dispose of the retransforming environment
```

Modifying the file MANIFEST.MF :
Manifest-Version: 1.0

```
Archiver-Version: Plexus Archiver
Created-By: Apache Maven
Built-By: XXXX
Build-Jdk: 1.6.0_43
Agent-Class: org.jboss.byteman.agent.Main
Can-Redefine-Classes: true
Can-Retransform-Classes: false
Premain-Class: org.jboss.byteman.agent.Main
```

setting true or false doesn't change anything for me.
as said :
https://www.ibm.com/developerworks/community/forums/html/topic?id=77777777-0000-0000-0000-000014774211
doesn't solve the problem for me with Attach API and IBM JDK .
I think it is due to the Install/Submit programm, because JDK IBM has extended the Attach API, and the attachment is a little different compared to HotSpot JVMs. I have succeeded to attach a JVM IBM and run the operation of the TraceByteman Mbean (flush / close Stream, activate/deactivate DO statement on rules)
Certainly Install / Submit must be modified to support IBM JVM.