Automated Performance Testing of SAP Process Orchestration

Git REPOSITORY TEST DOCUMENT

INT603

Exercises / Solutions  
Nicholas Holshouser / SAP Labs, LLC

Abhinag Palvadi / SAP Labs India Pvt. Ltd.

Andre Bakofen / SAP AG

**TABLE OF CONTENTS**

[1 Introduction 3](#_Toc459188748)

[2 Automated Testing 3](#_Toc459188749)

[3 Testing as a Service 3](#_Toc459188750)

# **Introduction**

This document is the master test document for TechEd CodeJam Session INT603. For automated tests the instructions are also maintained in the repository. This ensures all tests are run using the correct documents, tools, test plans, and reporting templates.

# Automated Testing

* Run the test as described in this document.
* If you find any problems or discrepancies please make your corrections and resubmit so these can be reviewed and (as needed) incorporated into the master branch

# Testing as a Service

It’s possible to automate even the infrastructure but in our case we only have an hour so the backend test systems are already provisioned and available. Based on your local laptop image name please pick your backend from the list below and use those configuration details.

Access Details for Backend

|  |  |  |
| --- | --- | --- |
| BACKEND SYSTEM | IP ADDRESS | CERTIFICATE |
| INT603\_BACKEND\_01 | 52.44.41.99 | INT603\_BACKEND\_01.ppk |
| INT603\_BACKEND\_02 | 52.21.146.162 | INT603\_BACKEND\_02.ppk |
| INT603\_BACKEND\_03 | 23.22.192.223 | INT603\_BACKEND\_03.ppk |
| INT603\_BACKEND\_04 | 52.45.173.17 | INT603\_BACKEND\_04.ppk |
| INT603\_BACKEND\_05 | 52.21.143.142 | INT603\_BACKEND\_05.ppk |
| INT603\_BACKEND\_06 | 52.45.201.11 | INT603\_BACKEND\_06.ppk |
| INT603\_BACKEND\_07 | 52.45.199.168 | INT603\_BACKEND\_07.ppk |
| INT603\_BACKEND\_08 | 52.23.71.239 | INT603\_BACKEND\_08.ppk |
| INT603\_BACKEND\_09 | 52.45.74.10 | INT603\_BACKEND\_09.ppk |
| INT603\_BACKEND\_10 | 52.45.203.71 | INT603\_BACKEND\_10.ppk |
| INT603\_BACKEND\_11 | 52.45.203.26 | INT603\_BACKEND\_11.ppk |
| INT603\_BACKEND\_12 | 52.45.147.180 | INT603\_BACKEND\_12.ppk |
| INT603\_BACKEND\_13 | 52.45.196.232 | INT603\_BACKEND\_13.ppk |
| INT603\_BACKEND\_14 | 52.45.167.252 | INT603\_BACKEND\_14.ppk |
| INT603\_BACKEND\_15 | 52.45.203.92 | INT603\_BACKEND\_15.ppk |

Access to the backend is using ‘root’ only and by certificate only

root: root user is accessible using certificate login only

The root user certificate is available in the repository

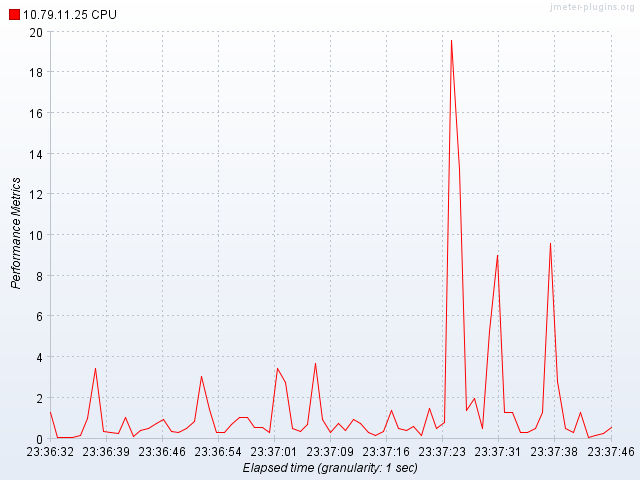
You can login to the backend (not necessary but possible) using Putty and your root certificate.

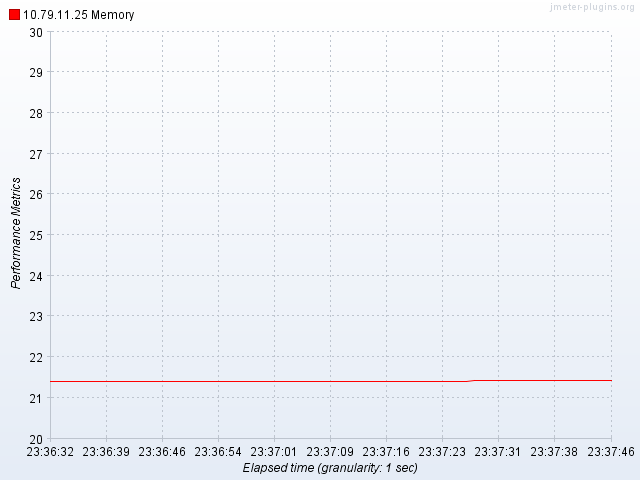
| **Explanation** | **Screenshot** |
| --- | --- |
| 1. Check your backend system is running and available:   http://<yourbackendIP>/nwa  User: Administrator  Password: Abcd1234 |  |
| 1. We’ll use Git bash for running scripts that execute the tests and start a backend performance agent   Start Git Bash from the Git Gui which has the advantage of opening the shell directly in your project directory  You will need two bash shells so start two.  JMeter uses a plug-in that connects to a running ServerAgent on the backend system and gets system performance metrics during the test. |  |
| 1. In Git Bash run the setup script   /c/INT603/test603/setup\_imageOWN.sh |  |
| 1. Check you connection to the server with the script serverconnect.sh   Pick the correct certificate for your server and copy/paste it (right-click copy and right-click paste) |  |
| 1. Enter the IP address |  |
| 1. You should see the result of the ‘df –h’ command executed on the backend |  |
| 1. Start the Performance Agent using the script startperfagent.sh and following the prompts   Enter the correct server certificate for your backend and the IP address of the server |  |
| 1. Run the warm-up script. This will make sure the application under test is warm and also check that the server is responding correctly.   In the bash shell running the performance agent you can see when the test script connects and begins gathering data. |  |
| 1. Check the results of the warm-up. You should have no errors.   You will also see the commands that generate graphs of the test results. These should also not have any errors. |  |
| 1. Check the output directory and verify the graphs have been generated.   The output directory is displayed in the bash shell as shown  You should have .png and .csv files |  |
| 1. Run the performance test using the script in Git Bash using the script Jmeter\_noguiOWN.sh   Follow the prompts by giving a test name. The results are stored in C:\INT603\test603>\csv\<SCENARIO NAME> |  |
| 1. Check the results |  |
| 1. Open NWA and check the SOA monitoring on backend   http://<ip>:50000/nwa  SOA => Monitoring => Performance Monitor  The overview shows us the aggregate data and we could derive our average message size (Total Size / Message Count)  We can also check the performance of different steps of the message processing.  In the example we can see that the processing time of the synchronous call – which matches the configured response time of the test web service  MS:module\_out:sap.com/com.sap.aii.af.soapadapter/XISOAPAdapterBean  By varying the message size (configurable in JMeter) we can see how size impacts performance |  |
| 1. Open PO Performance Monitors (in NWA) and check the detailed performance views   http://<ip>:50000/nwa  Availability and Performance => Resource Monitoring => Detailed Performance Statistics |  |
| 1. You can filter the Performance Statistics for the requests to the XISOAPAdapter   Action: /XISOAPAdapter/MessageServlet |  |
|  |  |
|  |  |

**Analysis**

Standard graphs generated from the test data are used for comparative analysis.

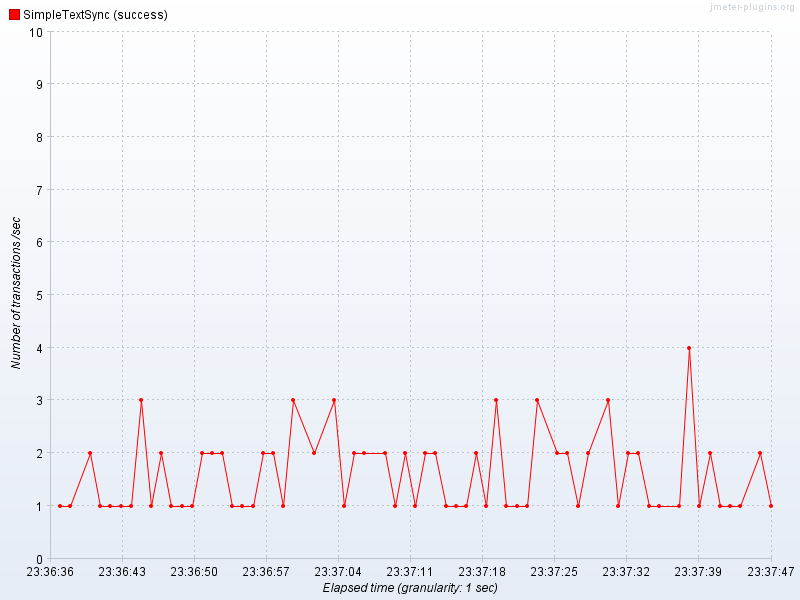
System Performance - CPU



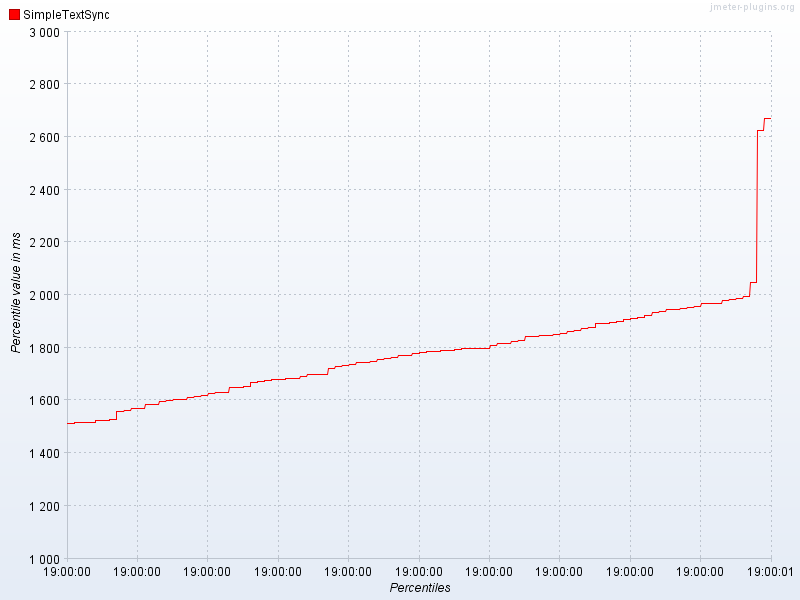


Test Results

Hits per Second (request rate)



Response Time Percentiles (useful for SLA analysis)



The NetWeaver performance monitor data and Detailed performance statistics are useful for analysing the unit performance and gathering the metrics such as unit cost of messages (bytes transferred, memory per request, CPU time, etc…

**Appendix**

We can also look at SYSTEMS as part of the repository and as we develop the system we can update master and then test the new feature

SYSTEMS can be standardized and then created dynamically in AWS. This includes the test client itself.

Take a vanilla system (Windows, Ubuntu, Linux) and make base test client

Test a standardized image of the application system (in our case it is a master image of a configured NW 7.5 Process Orchestration system with B2B and deployed content/scenario)

The standardized image can also be based on AWS and built dynamically using tools such as Cloud Foundry or Vagrant/Chef and EC2 API

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company. SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. Please see <http://www.sap.com/corporate-en/legal/copyright/index.epx#trademark> for additional trademark information and notices.

