# Neato SmartApps: Setting up Tsung Test Framework

Raja Software

Last updated: 08/18/2014

|  |  |  |
| --- | --- | --- |
| Date | Version | Author/Comments |
| 07/29/2014 | 0.1 | Initial draft |
| 08/06/2014 | 0.2 | Added verification steps for the Tsung set up. Also removed the footprint of test-7 from the document. |
| 08/07/2014 | 0.3 | Removed verification steps for jabber\_setup,xml and added information about how to 2 new test cases detailing how to customize it, run them and verify that they worked fine. |
| 08/14/2014 | 0.5 | Referred to Verify Setup, rewrote large pieces of description |
| 08/18/2014 | 0.6 | Added details on script to create robots, updated Customization section. |

## Introduction

Tsung (<http://tsung.erlang-projects.org/>) is a multi-protocol distributed load testing framework with which a real time heavy traffic scenario can be simulated to test the scalability and the performance of client/server applications. As part of Tsung framework, both HTTP and Ejabberd client libraries are provided. As the Neato SmartApps backend uses HTTP and XMPP extensively, Tsung is a good framework to test the Neato backend scalability and performance.

This document explains how to setup the Tsung framework and run some simple tests to verify everything is setup correctly. It also provides some information on how to run the various Neato backend specific tests, etc.

## Installation

Tsung can be installed using standard apt-get install but the source repositories are not pointing to the latest version, so it will install an older version of Tsung. We are using Tsung 1.5 as it has better Ejabberd client support. To install Tsung 1.5 on an Ubuntu machine, please fire following commands:

Execute following commands one by one

sudo apt-get update // This would update the source repositories

sudo apt-get install erlang // We need Erlang to install Tsung

sudo apt-get install make // This would install make

sudowget http://tsung.erlang-projects.org/dist/tsung-1.5.1.tar.gz

sudo tar -zxvf tsung-1.5.1.tar.gz // Untar the downloaded Tsung

cd tsung-1.5.1/

sudo ./configure

sudo make

sudo make install

Now if you simply execute **tsung** with not parameters, it will show you standard usage help.

## Setup Verification

We have provided 2 simple scripts to verify that Tsung is setup correctly – these are under the VerifyTsungSetup folder. Please see VerifyTsungSetup/VerifySetup.Readme.txt for more details.

### Test\_Setup\_1.xml

* This test makes an API request (getApiVersion) and prints the response
* The idea behind this test is to verify that your basic Tsung set up is working as expected.
* Search and replace “neatostaging.rajatogo.com” with your own hostname.
* To run, type “tsung -f Test\_Setup\_1.xml start”
* Expected Output:

HTTP/1.1 200 OK

...

Content-Length: 23

Content-Type: text/javascript; charset=UTF-8

{"status":0,"result":1}

* The most important part is the JSON result {"status":0,"result":1}. If you do not see this in the output, or you see other errors, it means that either your Tsung setup is incorrect, or the Neato backend server is not setup correctly.

### Test\_Setup\_2.xml

* This test brings 1 robot online and verifies if it came online.
* You need to customize the hostname, the robot serial number, chat id details and the Jabber hostname. See VerifySetup.Readme.txt for more details.
* To run, type “tsung -f Test\_Setup\_2.xml start”
* You should see the following JSON output in the log file

HTTP/1.1 200 OK

...

Content-Length: 73

Content-Type: text/javascript; charset=UTF-8

{"status":0,"result":{"online":true,"message":"Robot neato1 is online."}}

## Scalability Test Scripts – Overview

Once you have verified that the basic setup is working, you can start running other tests in the Scalability/Tsung folder (these folders have the format Test#). To get an overview of the tests, see Tsung\_Perf\_Test\_Overview.txt.

Each test typically has 2 Tsung scripts and a CSV file.

The XML script with jabber\_ prefix does the following:

* Brings XX number of robots online, sends initial presence packets to each robot, and then wait for X minutes to let the system stabilize.
* Once all the XX robots are online, the script sends http\_ping requests and IQ packets to the server (the periodic pinging keeps the connection alive and prevents eJabberD from shutting down the connection and causing the robot to go offline). Once all the pings are sent, there is a delay to cool off.
* Different variations of the script sends IQ or presence packet for ALL the robots in a given interval. Some scripts send the IQ/Presence packets and then wait for some time and then start all over again.

jusers.csv is used to authenticate the robots – it contains the robot serial numbers and passwords.

Get\_set.xml is another Tsung script that calls get and setProfile calls to all the robots. This script is run manually after all the robots are online (by running the first script).

## Increase “ulimit” value

Before you start running tests, you will likely want to increase the “ulimit” value – this controls the number of concurrent socket connections on your Tsung client. This value is set to 1024 by default, you can increase it by running

ulimit –n 10000

## Creating Robots in Bulk

Before you can run a test script to bring XX robots online, these robots have to be first created on the Neato SmartApps backend server. We have provided a simple server side PHP script that creates robots on the backend – this script is just a helper, and can be run before the actual test script itself.

IMPORTANT: This script needs to be run on the Neato backend server. You need to execute this command from /var/www/Neato\_Server/Server/Neato/protected/

The “jobToCreateRobots” takes 4 parameters:

* A robot prefix
* A numeric start index
* A numeric end index (greater than the start index)
* An optional email/user to link these robots against. Without this param, no link is established)

This test needs to be run from

To run Tsung script for scalability, system needs large number of robots already registered. To make this process easy, backend provides Yii PHP script. This script accepts 'prefix', 'start', 'end' and 'email'. Here parameters 'prefix', 'start' and 'end' are mandatory parameters where the 'email' is an optional parameter.

**Usage:**

php yiic robot jobToCreateRobots --prefix=value --start=value

--end=value [--email=]

**Example**

php yiic robot jobToCreateRobots --prefix=RSL --start=1 --end=5 --email=john@neatorobotics.com

This would create 5 robots with serial numbers RSL1, RSL2, RSL3, RSL4, RSL5 – each of these would be linked to a user whose email address is [john@neatorobotics.com](mailto:john@neatorobotics.com). The script also assigns predictable chat ids and passwords for each robot (if you were to create a robot via the standard interface the chat id is random number based on the current timestamp). For a robot RSL1, its chat id would be [robot\_RSL1@<ejabber\_host>.com](mailto:robot_RSL1@%3cejabber_host%3e.com) and the password would be “robot\_RSL1”. The “ejabber\_host” value depends on what the actual Ejabberd host value is specified in main.config.php on the Neato backend server.

## Customizing Test Scripts

The scripts assume that you have already created the necessary robots on the backend server. (Using the “jobToCreateRobots” will simplify the script configuration, but is not mandatory – you can create robots using any method).

* “UserArrivalRate” is the variable that decides how many robots would come online per second. This is used only in the script that brings robots online (starting with jabberd prefix).
* You can configure Ejabberd domain name – you need to change this in two places. The code snippet below uses neatostaging.rajatogo.com – and you need to replace it with your Ejabberd domain.

<servers><server host="[neatostaging.rajatogo.com](http://neatocloud1.rajatogo.com/)" port="5222" type="tcp"/></servers>

<options>

<option type="ts\_jabber" name="domain" value="[neatostaging.rajatogo.com](http://neatocloud1j.rajatogo.com/)"></option>

</options>

* To change the backend server, you can change the URL in the request section of the XML. . In the snippet below, the URL is “neatostaging.rajatogo.com” and needs to be replaced. The “subst” keyword indicates that the robot serial number will be replaced – you can use this in conjunction with the script to create robots (with a prefix and a serial counter).

<request subst="true">  
<http  url="<http://neatostaging.rajatogo.com/api/rest/json/?method=robot.get_profile_details2>"  
method="POST" version="1.1"contents="api\_key=1e26686d806d82144a71ea9a99d1b3169adaad917&amp;serial\_number=neato%%\_serial\_number%%"></http>  
</request>

* Each robot has its own eJabber chat id. So, if you want to deal with a range of chat ids, you can specify a prefix and the end count (the start is assumed to be 1). In the snippet below, the userid\_max value is 10 (indicating end counter is 10), and the eJabberD domain is “neatostaging.rajatogo.com” and the chat id prefix and password prefix is “neato”. (Note: in the script, it may have been more readable to call it “chatid\_prefix” and “password\_prefix” instead of “username” and “passwd”, respectively. The script below will do the actions for chat ids “neato1, neato2, … neato10”.

<options>

<option type="ts\_jabber" name="userid\_max" value="10"></option>

<option type="ts\_jabber" name="domain" value="[neatostaging.rajatogo.com](http://neatocloud1j.rajatogo.com/)"></option>

<option type="ts\_jabber" name="username" value="neato"></option>

<option type="ts\_jabber" name="passwd" value="neato"></option>

</options>

* You can also specify a series of robot chat ids and passwords using a CSV file (the csv file should be in the same directory as the script itself). See jusers.csv for a sample format. The script refers to this csv file as shown in the following snippet.

<options>

<option type="ts\_jabber" name="domain" value = "neatodev.rajatogo.com"></option>

<option name="file\_server" id='userdb' value = "**/home/Test6/jusers.csv**"/>

</options>

## Measuring Server Health, Misc. Tips

Typically you would run two Tsung scripts from 2 separate client machines (we were unable to run them on the same machine). For example, you would run the first script to bring all the robots online, and a second script to get/set robot data from another machine.

To collect server health information (reported in the Excel spreadsheet measurements), you would run the following commands on the Neato backend server.

pgrep -c apache2 // To see the total number of Apache processes

top // To get Memory and CPU usage of the MySQL, RabbitMQ and Ejabberd

lsof | grep 19181 | wc -l // to get the open file handles.

If your Neato backend server is a spread across 3 servers (Apache, MySQL and eJabber), you will need to run the above commands on all 3 servers. The key processes involved are: apache2 (owner www-data), mysql (owner root), beam.smp (with owner ejabberd) and beam.smp (owner rabbitmq-server).

To get more details about backend processes like robot log in, initial presence packets, IQ packets or check any error related to Ejabberd, check the log file called **ejabberd.log** which is located at **/var/log/ejabberd/** on the SmartApps backend server.

You can also check the Tsung log on the client machine (from where you are running Tsung performance script). These files are generated in the Tsung log directory – when you run a script Tsung will print a location to the runtime log file – you can check this log if there were any failures.

End.