Configuring LSGiWebConsole

Author: Janneth Rivera, Tere Gonzalez

Creation date: July 11, 2016

Last modification date: September 16, 2016

Contents

[**Overview** 1](#_Toc461801639)

[**1.** **Configuring WebServices - LSGiWebService package** 1](#_Toc461801640)

[Initial Requirements 1](#_Toc461801641)

[Configuration files 2](#_Toc461801642)

[a. environments.json 2](#_Toc461801643)

[b. datasets.json 3](#_Toc461801644)

[c. nodes.json 4](#_Toc461801645)

[d. lsgi.properties 5](#_Toc461801646)

[e. ssh.properties 5](#_Toc461801647)

[**2.** **Configuring UI - LSGiWebConsole package** 7](#_Toc461801648)

[Initial requirements 7](#_Toc461801649)

[Configuration files 7](#_Toc461801650)

[a. app.js 7](#_Toc461801651)

[**3.** **How to deploy pre-compiled LSGiWebConsole in Tomcat** 7](#_Toc461801652)

# **Overview**

This document describes the configuration steps to deploy LSGi Web services and LSGi Console. LSGi Web services are restful services that enable launch, stop and track inference jobs across multiple execution environments, datasets and computing nodes. LSGi Web Console is a user-friendly operational tool that uses LSGi Web services in order to monitor inference jobs tracking performance and results by iteration.

# **Configuring WebServices - LSGiWebService package**

## Initial Requirements

1. Code package: LSGi/source/webConsole/LSGiWebService  
   <https://github.hpe.com/labs/LSGi/tree/master/LSGi/source/webConsole/LSGiWebService>
2. Java 7 (version 1.7.0\_51) or later
3. Eclipse Java EE IDE, Version: Mars.2 Release (4.5.2)
4. Maven Integration for Eclipse plugin (m2e), version 3.3.3 or later
5. Spring Framework, version 3.2.3.RELEASE
6. Jackson library (JSON), version 1.9.13
7. Apache Commons Lang (Serialize/deserialize), version3.4
8. JSch (Java SSH), version 0.1.53
9. Log4J library, version 1.2.17
10. JClient.jar

## Configuration files

In order to work together with the LSGi engine, we need to edit a few configuration files first. These files are located under **LSGiWebService/src/main/resources/properties/** folder:

### environments.json

This file contains the environments that the LSGiWebConsole can run. In this file we need to specify the following information in JSON format for each environment:

* + id: unique id for this environment
  + name: This is the name of the environment, as will appear in the UI menu
  + hostname: This is the environment’s hostname or ip address to connect to by SSH
  + image: This is the icon’s image file name. (Image files reside in UI under /images folder)
  + lsgi\_home: This is the path where the LSGi package is located
  + description: This is the description shown in the UI menu for this environment
  + availableDatasets: This is a list of ids of the available datasets for this specific environment
  + availableNodes: This is a list of ids of the available nodes for this specific environment

Example:

[{

"id": 0,

"name": "Superdome X",

"hostname": "smaug-1.u.labs.hpecorp.net",

"image": "virtual-machine.svg",

"lsgi\_home": "/home/gomariat/graphs/code/LSGi\_BSIM\_LOOP/LSGi/LSGi\_nolibrarian",

"description": "The Superdome X environment has 16 NUMA sockets and 240 cores. LSGi engine runs multiprocesses and uses shared memory (tmpfs) to store shared states for the inference computation. The graphs are loaded in memory and ready to execute the inference.",

"availableDatasets": [0,1,2,3],

"availableNodes": [4]

},{

"id": 2,

"name": "Single Node-L4TM",

"hostname": "build-l4tm-2.u.labs.hpecorp.net",

"image": "Servers.png",

"lsgi\_home": "/home/gomariat/graphs/code/LSGi",

"description": "The l4tm-selfhosted-2 is a single node (DL580) running L4TM and librarian self hosted version.",

"availableDatasets": [0,1,2,3,4],

"availableNodes": [0,1,2,3,4,5,6]

}]

### datasets.json

This file contains the datasets that the LSGiWebConsole can run into the different environments. In this file we need to specify the following information in JSON format for each dataset:

* + id: unique id for this dataset
  + name: This is the name of the dataset, as will appear in the UI menu
  + value: This is the dataset path
  + image: This is the icon’s image file name. (Image files reside in UI under /images folder)
  + pollingTime: This is the time (in milliseconds) the UI is going to use to poll inference results from the Query Service . The UI will split this value depending on the number of nodes used.   
    For example: For 1 billion dataset, the UI is going to poll every:
    1. 3000 ms, when running 16 nodes
    2. 6000 ms, when running 8 nodes
    3. 24000 ms, when running 2 nodes and so on
  + loadingTime: This indicates the time (in milliseconds) we need to wait for this dataset to be loaded in the nodes before the inference starts
  + description: This is the description shown in the UI menu for this dataset
  + graphlabTime: This is the time GraphLab takes per iteration to run this dataset
    1. -1 indicates that GraphLab failed to run this dataset
  + queryServicePort: This is the port to connect with QueryService for this specific dataset
  + statistics: This are the graph statistics of this dataset
    1. vertices: Number of vertices in the graph
    2. edges: Number of edges in the graph
    3. s0: Number of vertices labeled as state0
    4. s1: Number of vertices labeled as state1
    5. labels: Total number of labels

Example:

[{

"id": 0,

"name": "1 Billion Vertices",

"value": "/data/inputGraphs/largeDataset/gV1000M.bin",

"image": "Big\_data.png",

"pollingTime": 48000,

"loadingTime": 4000,

"description": "The 1 billion vertex graph with 3.3 billion edges.. More details about the dataset.",

"graphlabTime": -1,

"queryServicePort": 58000,

"statistics": {

"vertices": 1000000000,

"edges": 3392337264,

"s0": 429382,

"s1": 278726,

"labels": 708108

}

},{

"id": 1,

"name": "200 Million Vertices",

"value": "/data/inputGraphs/largeDataset/gV200M.bin",

"image": "resources.svg",

"pollingTime": 16000,

"loadingTime": 2000,

"description": "The 200 million dataset with 6O0 million edges.. More details about the dataset.",

"graphlabTime": 77,

"queryServicePort": 58001,

"statistics": {

"vertices": 200000000,

"edges": 662942059,

"s0": 429382,

"s1": 278726,

"labels": 708108

}

}]

### nodes.json

This file contains the datasets that the LSGiWebConsole can run into the different environments. In this file we need to specify the following information in JSON format for each node option:

* + id: unique id for this node
  + name: This is the name of the nodes, as will appear in the UI menu
  + value: This indicates the number of nodes to use
  + image: This is the icon’s image file name. (Image files reside in UI under /images folder)
  + description: This is the description shown in the UI menu for this node

Example:

[{

"id": 0,

"name": "1 node",

"value": "1",

"image": "Infrastructure.png",

"description": "The 1 node... More details about the nodes."

},{

"id": 1,

"name": "2 nodes",

"value": "2",

"image": "Infrastructure.png",

"description": "The 2 nodes... More details about the nodes."

}]

### lsgi.properties

This file contains properties needed to execute launch/stop commands as well as retrieve results from the Query Service. In this file we need to specify the following information:

* + demoScriptsPath: This is the relative path in LSGi package to the demo scripts
  + launchScript: This is the name of the “launch” script in the LSGi package
  + stopScript: This is the name of the “stop” script in the LSGi package
  + tunnelPort: [Optional]. In case a tunnel is needed to connect to one of the environments, this indicates the port used to make the tunnel. (We use this to connect from our tomcat server to our QEMU VM master node <vm1-l4tm> because it does not have a public IP)

Example:

demoScriptsPath=/demo/inference

launchScript=launchMultiNodeDemo.sh

stopScript=killInferenceAll.sh

tunnelPort=9000

### ssh.properties

This file contains properties needed to create an SSH passwordless connection between the LSGiWebConsole and the execution environments. In this file we need to specify the following information:

* + user: This indicates the user to be use in order to execute the LSGiWebConsole commands
  + private\_key: This is the user’s private key used to authenticate. The value of this property correspond to the content of the file located at: “/home/user/.ssh/id\_rsa”. When copying the private key into this properties file, we need to append “\n\” at the end of each line.

Example:

user=johndoe

privateKey=-----BEGIN RSA PRIVATE KEY-----\n\

1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j\n\

1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j\n\

1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j\n\

1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j\n\

1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j\n\

1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j\n\

1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j\n\

1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j\n\

1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j\n\

1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j\n\

1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j\n\

1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j\n\

1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j\n\

1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j\n\

1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j\n\

1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j\n\

1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j\n\

1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j\n\

1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j\n\

1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j\n\

1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j\n\

1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j\n\

1a2b3c4d5e6f7g8h9i0j1a2b3c4d5e6f7g8h9i0j=\n\

-----END RSA PRIVATE KEY-----

**NOTE:**

SSH Passwordless **must be configured previously** from the server running Tomcat to each environment server.

For example:

In Tomcat server, with the user provided, you must be able to access each environment executing the following command:

**ssh <hostname | IP >**  
ssh build-l4tm-2.u.labs.hpecorp.net  
ssh 16.111.57.173

# **Configuring UI - LSGiWebConsole package**

## Initial requirements

1. Code package: LSGi/source/webConsole/LSGiWebConsole

<https://github.hpe.com/labs/LSGi/tree/master/LSGi/source/webConsole/LSGiWebConsole>

1. Java 7 (version 1.7.0\_51) or later
2. Eclipse Java EE IDE, Version: Mars.2 Release (4.5.2)
3. Bootstrap Framework, version 3.3.6
4. AngularJS Framework, version 1.0.7
5. D3 Framework, version 3.2.6
6. jQuery Framework, version 2.0.1
7. Underscore Framework, version 1.4.4

## Configuration files

There is only one file to configure for the UI. This file is located under **LSGiWebConsole/js/** folder:

### app.js

Update the variable “$rootScope.serverURL” with the URL where the Web Services reside. Follow the format:  
 **http[s]://<hostname | IP address>:<port>/<packageName>**   
  
Example:  
 $rootScope.serverURL = 'http://localhost:8080/LSGiWebService';  
 $rootScope.serverURL = 'https://myserver.mydomain.com/LSGiWebService';

//Production server:

$rootScope.serverURL = 'http://mercoop-26.hpl.hp.com/LSGiWebService';

# **How to deploy pre-compiled LSGiWebConsole in Tomcat**

1. Download the “**LSGiWebConsole.war**” and “**LSGiWebService.war**” files from our GitHub repository at:

<https://github.hpe.com/labs/LSGi/tree/master/LSGi/source/webConsole/LSGiWebConsole/dist>

<https://github.hpe.com/labs/LSGi/tree/master/LSGi/source/webConsole/LSGiWebService/dist>

1. Copy both WARs files in your Tomcat web server under **TOMCAT\_HOME/webapps** folder.
2. Start tomcat
3. Make sure configurations explained in previous sections were made.
4. Access the UI through the following URL:

**http[s]://<hostname | IP address>:<port>/<UIpackageName>**

Example:

<http://mercoop-26.hpl.hp.com/LSGiWebConsole>

If everything works fine, you should be able to see the following screenshot.

