Hackathon Lite Environment Setup Guide

About OpenTravel:

The OpenTravel Alliance provides a community where companies in the electronic distribution supply chain work together to create an accepted structure for electronic messages, enabling suppliers and distributors to speak the same interoperability language, trading partner to trading partner. Tens of thousands of the OpenTravel message structures are in use, carrying tens of millions of messages between trading partners every day.

We were founded in 1999 as a not-for-profit trade association by travel companies from airlines, car rental companies, hotels, cruise lines, railways, leisure suppliers, service providers, tour operators, travel agencies, solutions providers, technology companies and distributors.

As a not-for-profit trade association of travel companies, our primary focus remains the creation of electronic message structures to facilitate communication between the disparate systems in the global travel industry.

Members do the work of identifying what messages are needed, prioritize the work and collaborate to create the messages. Contact us at info@opentravel.org

Table of Contents

1 Introduction

OpenTravel's first Hackathon Lite will take place at the 2016 Advisory Forum in Orlando, Florida. The event will demonstrate the power of model-first design as a means of accelerating the delivery of new travel products to the marketplace. Participants will design and publish RESTful API specifications as mock services and compete to create the most innovative travel solution.

The Hackathon Lite development environment is comprised of several open source components and systems which work together as a self-service environment for creating RESTful mock services. Since it is composed entirely of free open-source components, it is readily available to all OpenTravel members. This document provides detailed instructions for those members who wish to setup similar environments for their own operations.

Although this document attempts to provide extensive details on the system setup and configuration for the Hackathon Lite environment, it will be helpful for readers to already be familiar with DevOps and J2EE technology, as well as some knowledge Linux system administration.

2 Environment Overview

Figure 1 below provides an overview of the components and lines of communication that exist within the Hackathon Lite development environment.

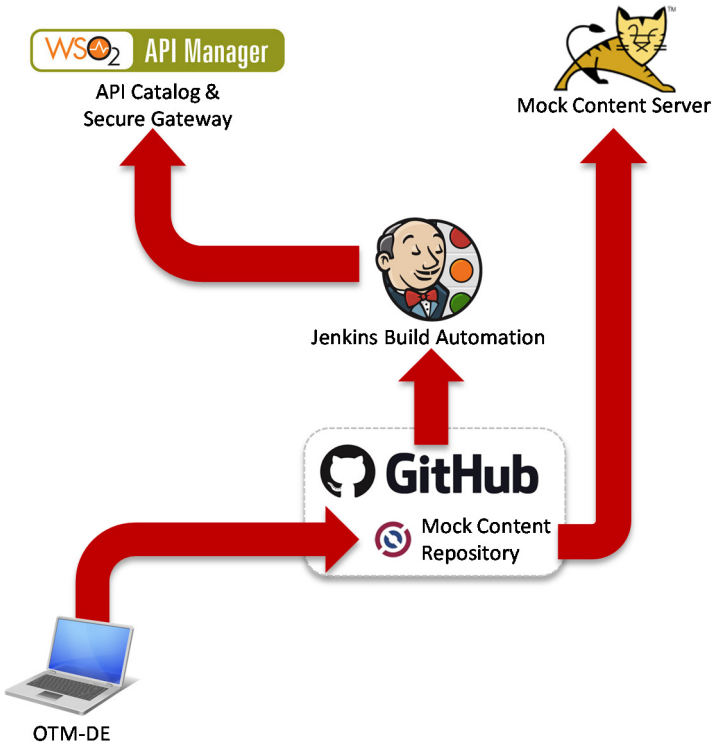

Figure 1: Hackathon Lite Environment Overview

The major components of the environment include:

- OTM-DE – Designer application for modeling information and API's using OTA2.0 technology
- GitHub Repository – Maintains OTM projects and mock content for all participants
- Jenkins – Coordinates publication of OTM resources to the API Catalog & Secure Gateway
- Mock Content Server – Tomcat web service to returns mock responses in response to live API calls
- WSO2 API Manager – Open source API Catalog and Secure Service Gateway

3 Setup Instructions

The following sections provide detailed setup instructions for the various run-time components of the Hackathon Lite environment.

3.1 GitHub Repository Setup

Source code and mock content repositories for the Hackathon lite can be found in the OpenTravel-Forum-2016 organization on GitHub. The organization consists of the following two repositories:

- otaforum-tooling – Tooling source code used to generate mock content files on the participants' local workstations, publish API's to the WSO2 API Store, and make the mock service content available via a live web service application. All source code is available for free download and use under the Apache 2.0 license agreement.

- otaforum-mock-content – Repository used to store and manage the mock service responses created by the Hackathon Lite participants. Section of this document describes additional configuration steps that will cause this mock content repository to be published automatically anytime participants push new updates to GitHub.

3.2 Virtual Machine Configuration

Other than the GitHub repositories, all of the software components used for the Hackathon Lite are deployed on a single cloud-based virtual server. This server runs on a Linux CentOS 6.7 operating system. While installations are possible using other versions and releases of the Linux OS, the instructions provided in this document are tailored to the CentOS platform.

Prior to installing the other 3rd party software applications, the following preparations and updates should be performed. Note that root access to the server VM is likely to be required to perform these and the other functions described in this document.

1. Install Java (version 7 or later) by running the following from a command window:

```
$ sudo yum install java-1.7.0-openjdk
```

2. Install the Git command-line client:

```
$ sudo yum install git
```

3. Register an internet domain name for the server. For the Hackathon Lite, a domain name (www.opentravelforum.net) was purchased and automatically registered with the VM server. For

instructions on how to perform this task in your own environment, contact your cloud hosting provider or network system administrator.

4. Create or purchase an SSL certificate for your domain name. This is necessary to secure the Tomcat application server and the WSO2 gateway (see Section 3.6). Certificates for the Hackathon Lite environment were delivered as separate certificate (.cert) and private key (.key) files.

3.3 Converting Certificates to JKS Keystores

Since both Tomcat and WSO2 are Java-based applications, SSL security is performed via JKS keystores. If certificates were delivered as separate certificate/key files as described in the previous section, the following steps must be performed in order to import those certificates into a JKS keystore. Note that the following commands require OpenSSL which is typically available on most Linux distributions.

1. Import the private key and certificate into a PKCS12 keystore by running the following OpenSSL command. You will be prompted to create a keystore password.

```
$ openssl pkcs12 -export -in <certificate-filename> -inkey
  <private-key-filename> -certfile <certificate-filename> -name
  tomcat -out keystore.p12
```

2. Use the Java keytool utility to import the PKCS12 keystore into a Java JKS keystore. During this step, you will be prompted to enter your password for the PKCS12 keystore and create a new password for the JKS keystore.

```
$ keytool -importkeystore -srckeystore keystore.p12 -srcstoretype
  pkcs12 -destkeystore keystore.jks -deststoretype JKS
```

The result of running these commands will be a Java keystore file named `keystore.jks` that contains your certificate and private key. The certificate will be aliased with the name "tomcat" as specified in the first OpenSSL command above.

3.4 Jenkins Installation and Setup

Jenkins is a DevOps continuous integration (CI) tool that provides an easy method for configuring, launching, and monitoring automated software builds. The Hackathon Lite environment uses Jenkins to perform the tooling builds and publish mock API's to the WSO2 API Store.

Installation instructions for Jenkins can be found here for several RPM-based Linux distributions such as CentOS. The details of configuring a Jenkins server are beyond the scope of this document, but user authentication must be enabled if remote build triggers are to be enabled via GitHub webhooks. For the Hackathon Lite environment, the following plugins must be installed at a minimum:

- Git Plugin (2.4.4 or later)
- Git Client Plugin (1.19.6 or later)
- Maven Integration Plugin (2.12.1 or later)

Additionally, the Jenkins system should be configured with a Git client installation and a Maven 3.2 Installation. This can be performed by following the links from the Jenkins home page to `Manage Jenkins → Configure System`. *NOTE: Do not use Maven 3.3 or later due to known issues with the OTM-DE-Compiler build*). The figures below show the correct settings for the Maven and Git installations on the Jenkins server.

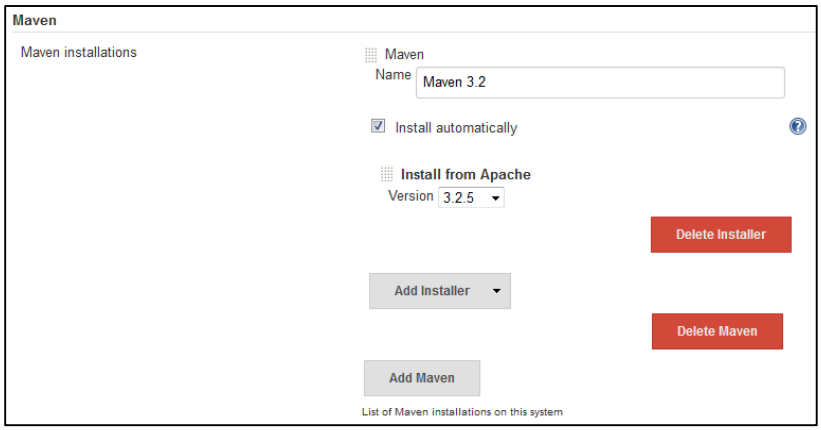

Figure 2: Maven Installation for Jenkins

Figure 3: Git Installation for Jenkins

Once Jenkins is installed and configured, the following build projects should be configured on the server:

OTM-DE-Compiler

Field Name	Value
Item Name/Maven Project Name	OTM-DE-Compiler
Project Type	Maven Project
Source Code Management	Select "Git"
Git Repository URL	https://github.com/OpenTravel/OTM-DE-Compiler.git
Git Branch Specifier	*/master
Build Triggers	None selected
Build: Root POM	pom.xml
Build: Goals & Options	clean install -DskipTests=true

otaforum-tooling

Field Name	Value
Item Name/Maven Project Name	OTA-Forum-Tooling
Project Type	Maven Project
Source Code Management	Select "Git"
Git Repository URL	https://github.com/OpenTravel-Forum-2016/otaforum-tooling.git
Git Branch Specifier	*/master
Build Triggers	Select "Build whenever a SNAPSHOT dependency is built"
Build: Root POM	pom.xml
Build: Goals & Options	clean install -DskipTests=true

otaforum-mock-content

Field Name	Value
Item Name/Maven Project Name	OTA-Forum-Mock-Content
Project Type	Maven Project
Source Code Management	Select "Git"
Git Repository URL	https://github.com/OpenTravel-Forum-2016/otaforum-mock-content.git
Git Branch Specifier	*/master
Build Triggers	Select "Trigger builds remotely (e.g., from scripts)"
Authentication Token	de12233b-3cf0-41b5-a135-e623e8b58b48
Build: Root POM	pom.xml
Build: Goals & Options	install

After these builds are configured, the 'OTM-DE-Compiler' build should be executed. Once complete, this should automatically run the 'OTA-Forum-Tooling' build as well.

3.5 Tomcat Installation and Setup

Apache Tomcat is a very popular open source J2EE web application server. To download and install on your Linux VM, run the following commands in the directory where you want to the server's software to be installed:

```
$ wget http://apache.mirrors.tds.net/tomcat/tomcat-
  8/v8.0.33/bin/apache-tomcat-8.0.33.tar.gz
$ tar -zxvf apache-tomcat-8.0.33.tar.gz
```

After running these commands, the server should be unpacked in a directory named `/apache-tomcat-8.0.33`. For the remainder of this section, this installation directory will be referred to as `<tomcat-home>`. Tomcat comes pre-configured to run insecurely, so there are some additional steps to setup the SSL certificates and install the mock content web application.

1. Remove the default web applications that come bundled with the standard Tomcat distribution.

```
$ cd <tomcat-home>/webapps
$ rm -rf *
```

2. Download and copy the .war file from the Jenkins build of the OTA-Forum-Tooling project. This file can be found by following the links to the "Last Stable Build" of the project and clicking on the

link for the "Mock Content Web Application" project. Once the .war file has been downloaded and copied to the `<tomcat-home>/webapps` folder, rename the file to `mock-content.war`.

3. Create a new file named `<tomcat-home>/conf/ota2-mockserver.properties` with the following content. Note that the first setting is the URL location of the `otaforum-mock-content` repository on GitHub.

```
org.opentravel.mockServer.remoteRepositoryUrl=https://github.com/OpenTravel-Forum-2016/otaforum-mock-content.git
org.opentravel.mockServer.localRepositoryPath=<tomcat-home>/git-repo
```

4. Next, we will configure the SSL certificate for use by the Tomcat server. Start by copying the `keystore.jks` file you created in Section 3.3 to the `<tomcat-home>/conf` directory.

5. Edit the `<tomcat-home>/conf/server.xml` file to reference the keystore file you copied in the last step by adding the following entry:

```
<Connector SSLEnabled="true" clientAuth="false"
  keystoreFile="<tomcat-home>/conf/keystore.jks"
  keystorePass="<your-keystore-password>" maxThreads="150"
  port="8443" protocol="org.apache.coyote.http11.Http11Protocol"
  scheme="https" secure="true" sslProtocol="TLS"/>
```

6. While still editing the `<tomcat-home>/conf/server.xml` file, make the following highlighted change to the non-secure connector entry. Save and close the file once this step is complete.

```
<Connector connectionTimeout="20000" enableLookups="false"
  port="8080" protocol="HTTP/1.1" redirectPort="8443"/>
```

7. Edit the `<tomcat-home>/conf/web.xml` file by adding the following highlighted entry just before the closing tag of the file. Save and close the file once this step is complete.

```
  <security-constraint>
    <web-resource-collection>
      <web-resource-name>Protected Context</web-resource-name>
      <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <user-data-constraint>
      <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
  </security-constraint>

</web-app>
```

8. Start the Tomcat server by running the following command:

```
$ <tomcat-home>/bin/startup.sh
```

3.6 WSO2 Installation and Setup

The WSO2 application provides an online API Catalog for the Hackathon Lite environment, as well as a gateway that provides SSL security and OAuth2 authentication for API consumers. To install the API Manager, first download the application zip archive from the WSO2 API Manager web site. Then upload or copy it to your server VM and unzip it with the following command:

```
$ unzip wso2am-1.10.0.zip
```

For the remainder of this section the directory where the WSO2 application is installed will be referred to as `<wso2-home>`. Like Tomcat, the WSO2 API Manager is configured to run out of the box. To operate properly within the Hackathon environment, however, several changes and configuration updates are still required.

3.6.1 Configuring the SSL Certificates

The secure gateway of the WSO2 API Manager utilizes two JKS keystores for SSL security. Both of these files are located in the directory `<wso2-home>/repository/resources/security`.

- `wso2carbon.jks` – Private key, certificates, and trust chains used for communication with external (client) connections through the gateway. By default, the password for this keystore is "wso2carbon".
- `client-truststore.jks` – Public keys used to connect with back-end applications which register API's with the WSO2 API Store

To start with, things will be easier if you copy your domain keystore (created in Section 3.3) to the same folder where these files are located. Once that is done, perform the following steps to add your certificates to the WSO2 keystores:

1. Export your domain's certificate and import it into the WSO2 client trust store with the following commands:

```
$ keytool -exportcert -keystore keystore.jks -storepass <password>
  -alias tomcat -file tomcat.cert

$ keytool -importcert -file tomcat.cert -alias tomcat -keystore
  client-truststore.jks -storepass wso2carbon
```

2. Import your (self-signed) private key entry into the WSO2 keystore:

```
$ keytool -importkeystore -srckeystore keystore.jks -srcstoretype
  jks -srcstorepass <password> -destkeystore wso2carbon.jks -
  deststoretype jks -deststorepass wso2carbon
```

3. Delete the original WSO2 private key and rename your domain's private key entry to effectively replace it:

```
$ keytool -delete -keystore wso2carbon.jks -storepass wso2carbon -
  alias wso2carbon
```

```
$ keytool -changealias -keystore wso2carbon.jks -storepass
  wso2carbon -alias tomcat -destalias wso2carbon -keypass
  <password>
```

4. Change the password of the WSO2 keystore to match that of your private key:

```
$ keytool -storepasswd -keystore wso2carbon.jks -storepass
  wso2carbon -new <password>
```

Now that the keystores have been configured with the proper private keys and certificates for your domain, several of the WSO2 configuration files must also be updated. In each of the following files, update the keystore password to match that of your domain's private key and certificate (hint, search for "jks" to find all of the places where the password needs to be updated):

- `<wso2-home>/repository/conf/carbon.xml`
- `<wso2-home>/repository/conf/axis2/axis2.xml`
- `<wso2-home>/repository/conf/tomcat/catalina-server.xml`
- `<wso2-home>/repository/conf/identity/identity.xml`
- `<wso2-home>/repository/conf/identity/EndpointConfig.properties`

In addition to updating the keystore password, the following files require additional changes:

- `carbon.xml` – Uncomment the `<HostName>` tag and update the value with your server VM's domain name.
- `axis2.xml` – In the `<transportSender>` section, uncomment the `HostNameVerifier` parameter and set the value to `AllowAll`

3.6.2 Verifying the Configuration

To verify the configuration of the server, first start the server and watch the logs during startup by running the following commands:

```
$ <wso2-home>/bin/wso2server.bat -start
$ tail -f <wso2-home>/repository/logs/wso2carbon.log
```

If you see any exceptions (Java stack traces) during startup, it typically means that you have missed an update or entered an incorrect keystore password. If no exceptions are thrown, proceed to the following steps:

1. Login to the API Publisher portal (see link in the Appendix) as the administrator (userid/password is 'admin/admin')
2. Click the "Add" button to add a new API and select the "Design a new API" option.
3. On the Design screen, enter the following values and click the "+ Add" button
   - Name: test
   - Context: test
   - Version: 1.0

- URL Pattern: /google (check the GET checkbox)

4. Click the "Next: Implement" button, select the "Managed API" button on the next screen, and enter the following values:
   - Endpoint Type: HTTP Endpoint
   - Production Endpoint: `https://<your-domain-name>:8443/mock-content`
5. Click the "Next: Manage" button, and select "Unlimited" for the Tier Availability value
6. Click the "Save and Publish" button. When prompted click the "Go To API Store" button
7. Login to the API Store and click the "Subscribe" button on the "test-1.0" API screen. *NOTE: An error at this step typically indicates a configuration problem in one of the identity files listed above.*
8. Click on the "API Console" tab, then click the "GET" button next to your API operation name. When you see the "Try it Out!" button, click it.
9. *Don't worry about the results of this API call. As long as you do not see SSL certificate errors, things are working fine.*

3.6.3 Changing the Administrator Password

The final configuration item that should be changed for WSO2 is the administrator password – unfortunately, this is not as straightforward as it sounds. To change the password, perform the following steps:

1. Open a browser window for the WSO2 API Manager system console (see appendix for the address).
2. Login using the administrator's default user ID and password (admin/admin).
3. Under the "Users and Roles" tab on the left side of the screen, click "List". Then click the "Change My Password" button.
4. Enter your old and new passwords, then click the "Change" button.
5. Now edit the file `<wso2-home>/repository/conf/user-mgt.xml`. In the `<AdminUser>` element change the password to the new one that you just entered on the system console.
6. Restart the WSO2 server for this change to take effect.

3.7 GitHub Webhook Configuration

The last configuration setting for the DevOps environment is to setup web hooks for the otaforum-mock-content repository on GitHub. On the GitHub web site for your mock content repository, click on the "Settings" link and select "Web Hooks and Services" from the tabs on the left side of the page. Then use the "Add Webhook" button to add the following web hook URL's:

1. Create a webhook that will cause the Mock Content web application (running on Tomcat) to refresh itself whenever user's commit new content to the repository:

```
https://opentravelforum.net:8443/mock-content/admin/refreshContent
```

2. Create a second webhook that will trigger a build on the Jenkins server to push OTM resource definitions to the WSO2 API Store:

```
http://opentravelforum.net:9090/jenkins/job/OTA-Forum-Mock-
  Content/build?token=de12233b-3cf0-41b5-a135-e623e8b58b48
```

NOTE: If the first web hook listed above (refresh-content) does not fire correctly for some reason, the same URL can be launched from a normal web browser. This will force the mock-content application on the Tomcat server to automatically pull all of the latest changes from GitHub.

3.8 Mock Content Repository Configuration

Because the mock content development process requires certain scripts to run from the user's workspace and the Jenkins build server, there is one configuration file which must be updated with the correct URL's for the various components of the Hackathon Lite environment. When setting up your own environment, the following files must be updated for your own environment.

- `.config/api-publisher.properties` – Update this file with the appropriate settings for your environment. The Appendix of this document contains URL locations for components deployed within the official Hackathon Lite environment.
- `.config/update-context-app.jar` – This executable Java archive is created by the OTA-Forum-Tooling Jenkins build. Whenever the tooling project is modified, a new copy of the Jar should be downloaded from the stable build link for the "Update Context Application" sub-project.

3.9 Developer Workstation Setup

The system requirements and setup instructions for developer workstations is covered in the Participant Webinar presentation for the Hackathon Lite. The presentation and webinar recording can be downloaded from the following links:

- Hackathon Lite Participant Webinar (Presentation)
- Hackathon Lite Participant Webinar (Recording)

4 Conclusion

While it is certainly still true that common standards within an industry can help to accelerate development and reduce the time to market for new solutions. With the introduction of new model-first design and development technologies such as the OTM-DE, however, the OpenTravel Alliance has moved beyond its traditional role as a clearinghouse for free XML schemas. By expanding and enhancing those capabilities with 21st century DevOps tools and technology, the OpenTravel Alliance hopes that the Hackathon Lite event will serve to inspire innovation among its members and within the travel industry at large.

Appendix: Hackathon Lite Application Links

This section provides a consolidated list of the links and URL's for the various components of the OpenTravel Hackathon Lite environment.

GitHub Organization:	https://github.com/OpenTravel-Forum-2016
Jenkins Dashboard:	http://opentravelforum.net:9090/jenkins
WSO2 API Manager Console:	https://opentravelforum.net:9443/carbon/admin/index.jsp
WSO2 API Publisher:	https://opentravelforum.net:9443/publisher/
WSO2 API Store:	https://opentravelforum.net:9443/store/
Mock Content Server: *(Base URL)*	https://opentravelforum.net:8443/mock-content
Mock Content Server: *(Manual Refresh URL)*	https://opentravelforum.net:8443/mock-content/admin/refreshContent