

**Oracle with SSL-Example**

June 15, 2017

**Document Control Information**

**Document Information**

|  |  |
| --- | --- |
| **Name** | Oracle with SSL-Example |
| **Program Name** | Enterprise Integration |
| **Author** | Rohit Vyas |
| **Version** | 1.0 |
| **Status** | Draft |

**Document Edit History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | Date | Description | Author |
| 1.0 | 27/01/2017 | Initial | Rohit Vyas |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Support / SME Resources**

|  |  |  |  |
| --- | --- | --- | --- |
| **Team** | Name | Role | Contact (Email / Phone) |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Contents

[1. Overview 4](#_Toc485627735)

[2. Purpose 4](#_Toc485627736)

[3. Prerequisites for using example 4](#_Toc485627737)

[4. Contents of example 8](#_Toc485627738)

[5. Use Case 10](#_Toc485627739)

[6. POM details 10](#_Toc485627740)

[7. Important Notes 11](#_Toc485627741)

1. Overview

This document describes how we can connect to Oracle DB via SSL.

This is not a template but an example as a reference to be used for connecting to Oracle DB with SSL. The example is built on Mule Runtime 3.8.3 and JDK 1.8, but it will also work on Mule Runtime 3.8.x and higher.

1. Purpose

SSL enabled connection to Oracle DB Server requires lot of configuration to be done at client Oracle instance as well as server Oracle instance. We have done this set up once in one machine and got the unique SSO file (registered at server end), which can be used from any client machine to connect to Oracle Server.

This POC was initially done in Java and then successfully done with Mule generic DB connector.

So Instead of doing same activity for the development repeatedly, we are providing this example.

Developers can download this example and use it per their requirement. That will reduce the efforts of the developer.

1. Prerequisites for using example

Before using this example, some basic things developer needs to follow:

1. Install java and maven in local machine, details regarding configuration and installation of maven and java can be found in the below document link:

<https://github.com/ICOE-Templates/MuleSoft-ResuableTemplates/raw/master/Template-Docs/DeveloperMachineSetup_v1.1.docx>

1. Now we need to download and configure the anypoint studio. Details regarding this is

provided in the below document. It is important is install some plugins as well for anypoint studio, all the details have been provided in the below document link:

<https://github.com/ICOE-Templates/MuleSoft-ResuableTemplates/raw/master/Template-Docs/AnypointStudioSetup_v1.1.docx>

1. Download the example from Exchange in Anypoint Studio IDE.

Following steps can be followed for downloading example.

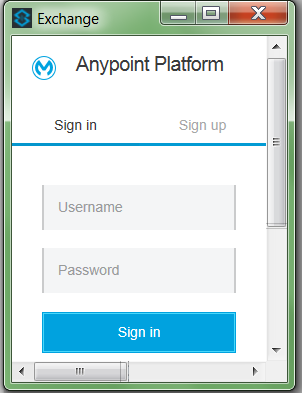
1. For this open Exchange in Anypoint Studio from the menu bar.



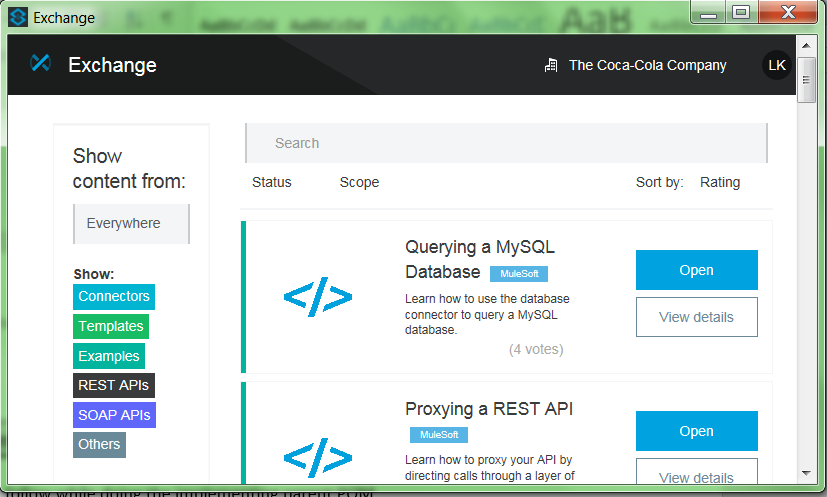
Colored symbol is the Exchange option.

(b) After selecting this option a new window will open and after loading it will ask for sign in

with Anypoint login and password

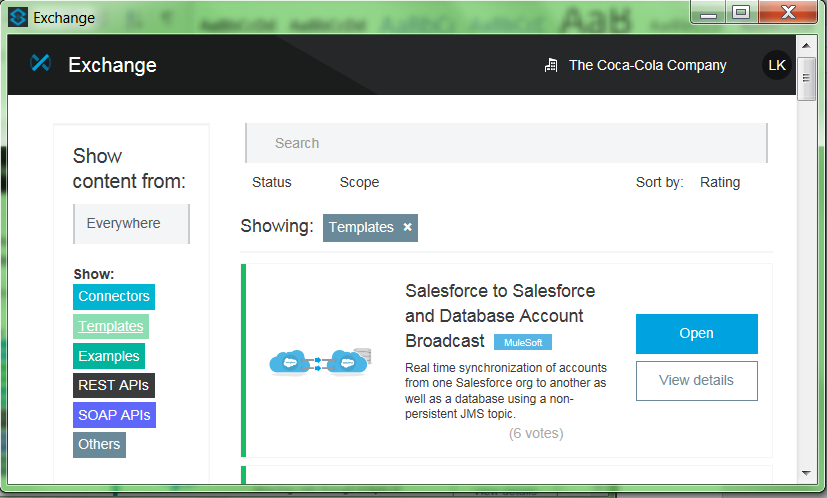


(c) After successful login, a new window will appear like below

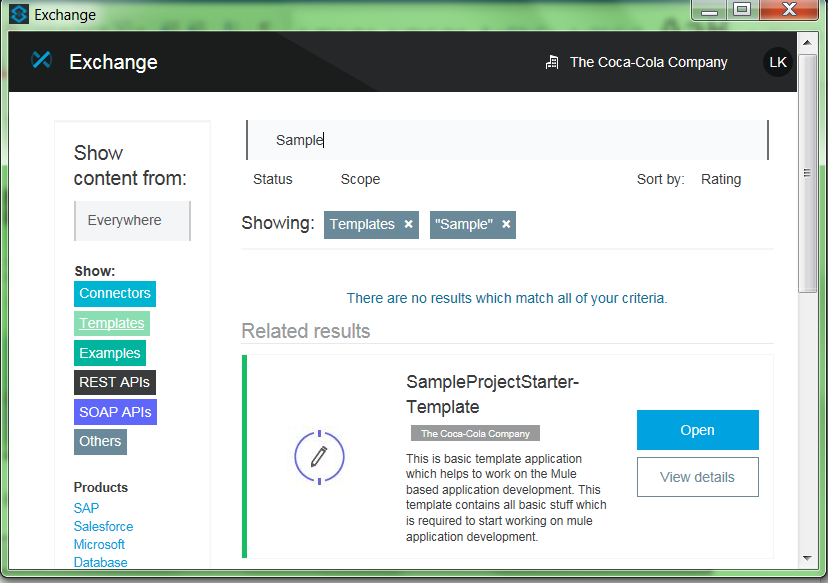


Change from here

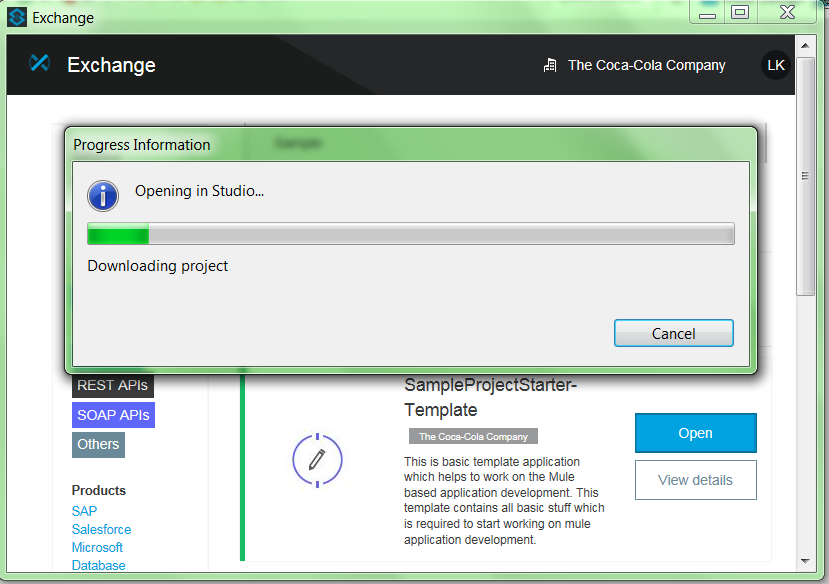
(d) In the present window please select Examples.



(e) Then type the name of Example in search box.

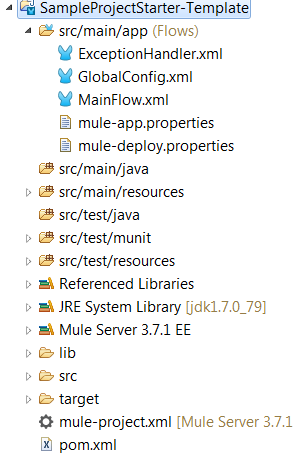


(f) Now click Open to download the “**OracleWithSSL-Example**”.



The project will download in the workspace of the Anypoint Studio.

(g) Now developer can start working on it.



1. Contents of example

This example has following things:

1. It is mavenized using parent-pom.xml and pom.xml.

For more details of parent-pom.xml and its uses please refer below document link.

<https://github.com/ICOE-Templates/MuleSoft-ResuableTemplates/raw/master/Template-Docs/UseOf-ParentPOM_v1.0.docx>

1. It contains logging-metadata file inside the resources folder which are required for implementing Logging Framework and contains loggingframework-2.0.1.jar. dependency in pom files

by using which maven will the jar from nexus repo.

For more details of Logging Framework, and its uses please refer below document link.

<https://github.com/ICOE-Templates/MuleSoft-ResuableTemplates/blob/master/Template-Docs/UseOf-LoggingFramework_v1.3.docx>

1. It contains environment based properties files (like dev/test/prod) inside the resources folder. Developer should rename these files before using it like <applicationName>-${env}.properties.

e.g. <applicationName>-dev.properties

<applicationName>-test.properties

<applicationName>-prod.properties

1. It contains secure-property-placeholder, to secure (encrypt) the key and some sensitive data which are available in the properties files. Dummy key is provided in the mule-app.properties.

Developer shouldn’t use the same key, it needs to change before using the example.

1. It contains Munit test files inside the munit folder, for providing the sample demo to

developers. So, this munit test cases should be created while working with mule flow.

1. It contains the sample mule flow to provide the demo for the developer by using http

component which needs to be invoked to connect to DB and get all results.

1. Exception handling flows which contains basic exception strategies. Developer can include more based on their requirement.
2. Global configuration file contains all global properties and configuration which are required for all mule configuration files. This example also includes health check framework.
3. The example provides extra layer of security for communication between the Mule ESB and the Oracle DB using SSL.
4. In order to use the health check framework the developer needs to add the below properties in the mule-app.properties file:

#Application health check properties

healthcheckservicedomain=tcccplatformframework-dev.cloudhub.io

healthcheckserviceport=443

healthchecklistenerhttpconfig=HTTP\_Listener\_Configuration

healthcheckbasepath=healthcheck

blockedcountthreshold=10

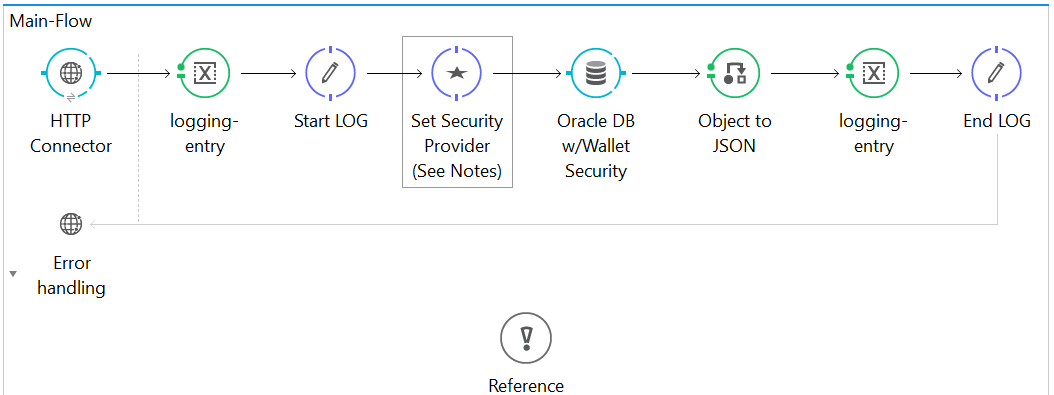
blockedtimemin=1000

#do regex here example below check for any class that has the word groovy or grizzly in them.

blockincludeclasses=(.\*?)

includeclasses=(.\*?)

1. Use Case



There is one Main Flow, in which we have a HTTP connector used to expose a web-service. This web-service when invoked connects to Oracle DB with SSL.

The Groovy component “SetSecurityProvider” is mandatory as we are adding oracle security config in runtime, as JAVA security will not be able to understand Oracle Security Config.

1. POM details

In Oracle with SSL example there are two dependencies that the example depends on, they are the two oracle jars:

* Ojdbc7
* Oraclepki

There are several other jars which oraclepki jar is dependent upon, which also needs to be added.

* ojpse
* osdt\_cert
* osdt\_core

1. Important Notes

Things which should be renamed in example before using it.

(1). Project Name

(2). Mule configuration file names

(3). Properties file names (except logging-metadata,json file because they are mandatory files

whose name shouldn’t be changed)

(4). Secure property key value (in the example it is given as dummy just for demo)

(5). Mule flow names which should be as application requirement.

(6). Munit flow names which should be as application requirement.

Consider the below details before sending a mail for deployment.

1. CR Number 2. Application Name 3. SVN Location

4. Properties for application

(a) env=dev (b) key=icoe!123

5. Deployment Details (Cloudhub)

(a) Worker Size = .1vCores (b) Workers = 1 (c) Mule Runtime = 3.7.1

6. For on Premise deployment

(a) Consider only above 1-4 options

(b) Make sure http component shouldn't as source in mule flow in the application.