**OTM-EDE Integration(Google)**

Specification Document

Version 1.1

**Inspirage Contacts**

Akshay Thakur

**Akshay.thakur@inspirage.com**

 +91 992-324-6991

Created: 30-Jun-2018

Updated: 02-July-2018

Contents

[1 Change History 3](#_Toc13735177)

[2 Reviewers 3](#_Toc13735178)

[3 Introduction 4](#_Toc13735179)

[4 Prerequisites 5](#_Toc13735180)

[5 Development 6](#_Toc13735181)

[5.1 Using JAVA Interface: 6](#_Toc13735182)

[5.2 Using WEB Service Interface: 7](#_Toc13735183)

[6 Configuration and Deployment 9](#_Toc13735184)

[6.1 Java Class 9](#_Toc13735185)

[6.2 Web Service 10](#_Toc13735186)

[7 SSL Configuration (Only for on-premise) 11](#_Toc13735187)

[8 Google GeoLocation API Key Generation 13](#_Toc13735188)

[9 Test Case 14](#_Toc13735189)

[10 Source Code 16](#_Toc13735190)

[11 Open & Closed Issues 17](#_Toc13735191)

# Change History

| Date | Author | Version | Change Reference |
| --- | --- | --- | --- |
|  |  |  |  |
| 30-Jun-2018 | Akshay Thakur | 1.0 | Initial |
| 02-July-2018 | Akshay Thakur | 1.1 | Support for Tomcat7+ version added |
|  |  |  |  |
|  |  |  |  |

# Reviewers

| Name | Position |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Introduction

The intent of this document is to provide development, configuration and deployment of Google distance engine which can be integrated with OTM application. OTM application has exposed some Java APIs that can be used by any third-party application to extend OTM functionality. These external APIs are stand-alone objects with a Java interface provided by Oracle Transportation Management to minimize the programming effort and maximize runtime performance.

This document will cover one of the API call used for OTM location address validation using EDE interface.

# Prerequisites

1. Java development environment (Eclipse or NetBeans) for building necessary implementation.
2. Additional third-party Jars needed to support above implementation. These Jars are mainly dependencies which are required to compile our classes. This may change based on developer’s interest.
3. SSL certificate from Google API, this can be downloaded from URL <https://maps.googleapis.com/maps/api/geocode/xml>
4. Google API Key should be created before using the API.
5. Need any OTM running instance either cloud or on-premise.

# Development

As per guidelines provided by OTM standard documentation (External Programming Interface Guide) with respective version, there are 2 ways we can develop this EDE callable interface.

Refer your OTM install directory to find External Programming Interface Guide:

**Path**: $GLOG\_HOME/docs/otm/integration

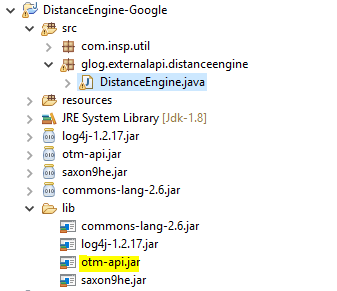
Before developing below interfaces, we need to get Google Geolocation API key, refer [Section 8](#_Google_GeoLocation_API)

## Using JAVA Interface:

This approach is only supported for OTM on-premise version because we need to load our custom Jar in OTM application server. OTM has provided the sample example for implementing Java interface for distance lookup, we can take advantage of same example and implement the business logic.

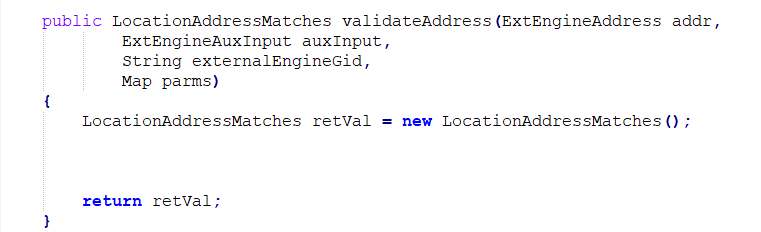
Below are some steps to be performed.

* Create Basic Java project in Eclipse or NetBeans IDE for development of this interface.
* Import OTM API library (otm-api.jar) into project which will be useful for implementation. Snapshot below,



This jar is located under directory: $GLOG\_HOME/externalAPI/lib

* Snapshot of method use for Development,



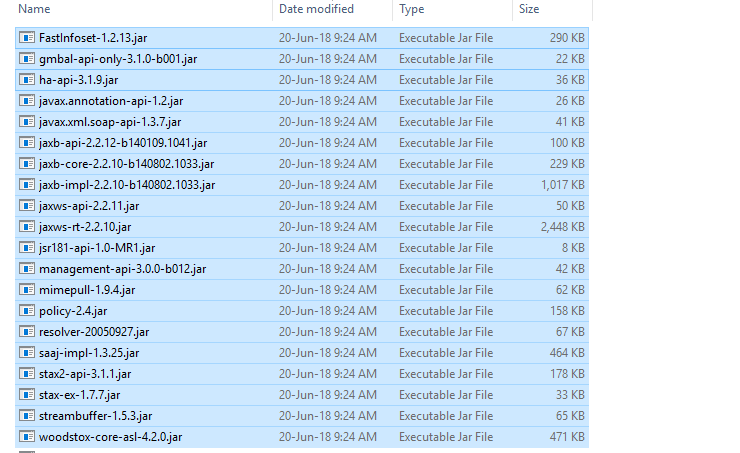
* Once done with completion of Project, we can export this Java project as Jar. This will be a final Jar file which needs to be uploaded on OTM application server.

## Using WEB Service Interface:

This approach is supported in both OTM on-premise and Cloud version. We need to configure our application web service (inline with OTM definition) in OTM application to invoke this integration. We can refer Web Service implementation section within External programming interface guide to develop our application.

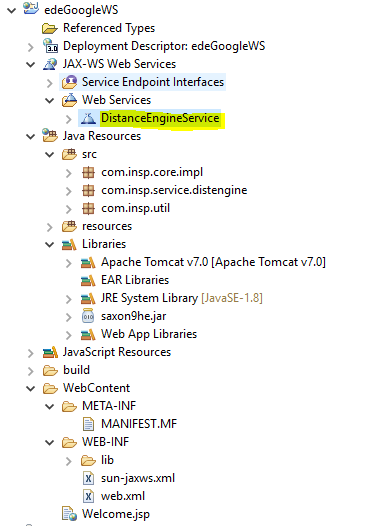
Below are some steps to be performed.

* Create Dynamic Web Project in Eclipse or NetBeans or Maven Project.
* Generate web services of application using WSDL provided by OTM within external programming guide.
* Generate Stubs using WSDL file.
* Do the service end point implementation.
* Compile and Package service artifacts.
* Deploy the webservice.
* Below are some dependencies (Jars) which are useful in Webservice development. These are related Jars for Web service project.



These Jars can be downloaded as JAX WS RI Runtime Bundle.

Project Snapshot,



* Once all the above steps are done, we need to export this project as WAR file and deploy it on tomcat application server.

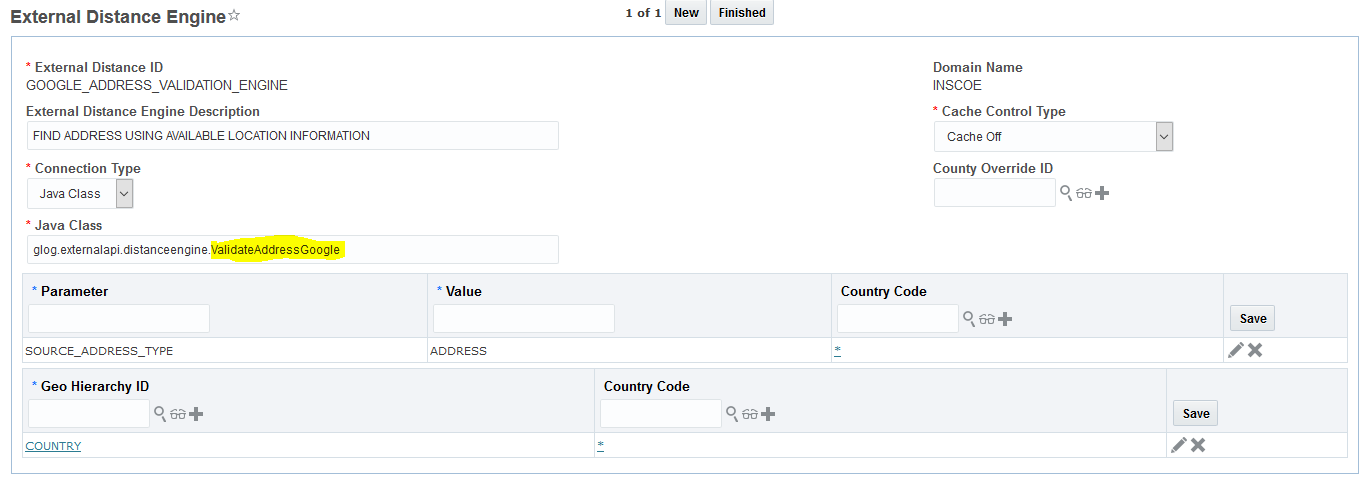
# Configuration and Deployment

This section provides the details of configuration to be done at OTM application and server side.

## Java Class

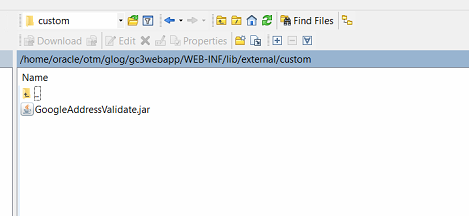
Steps to configure (This is only valid for OTM on-premise version)

* Create External Distance Engine in OTM with details mentioned in below screen shot,



Highlighted part is the main implementation class from custom Jar.

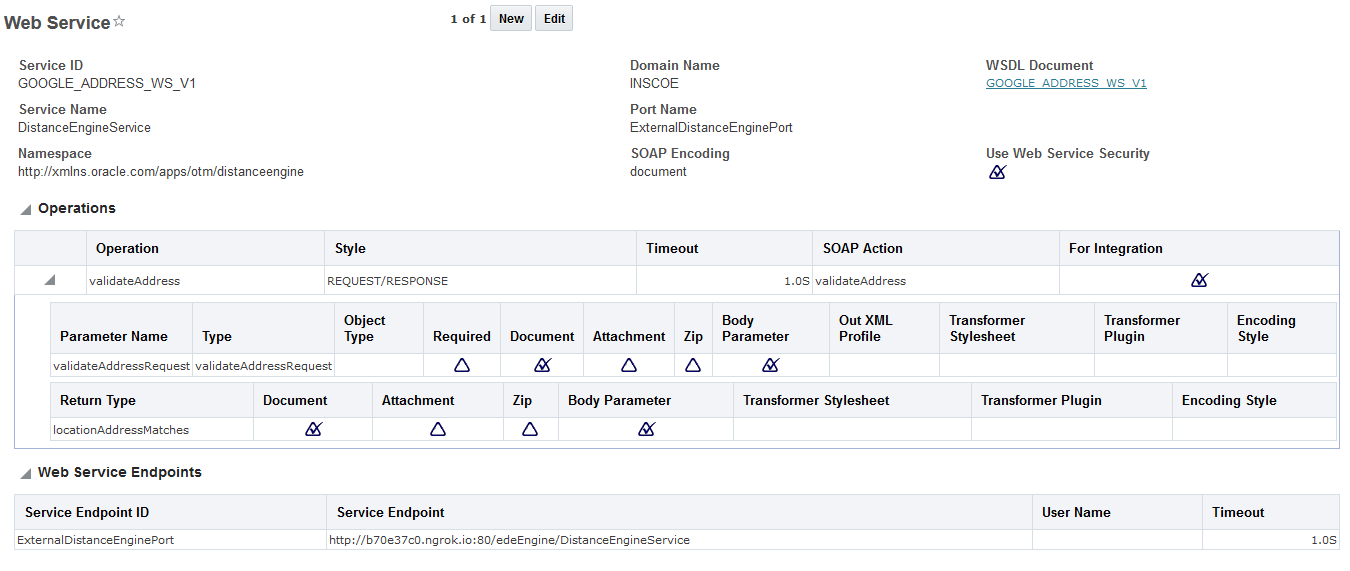
* Once Distance Engine is configured, we need to upload the custom Jar on OTM application server. Refer External Programming Interface Guide for respective OTM version, this is because OTM directory structure is getting changed based on versions.

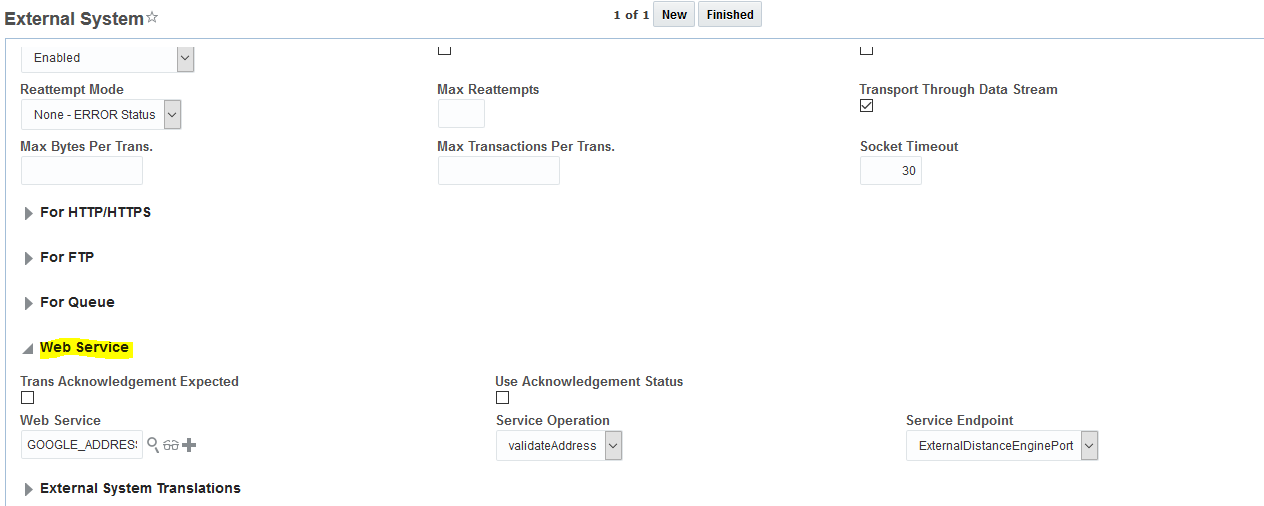


* Finally, restart OTM application to load this Jar in classes folder of OTM.

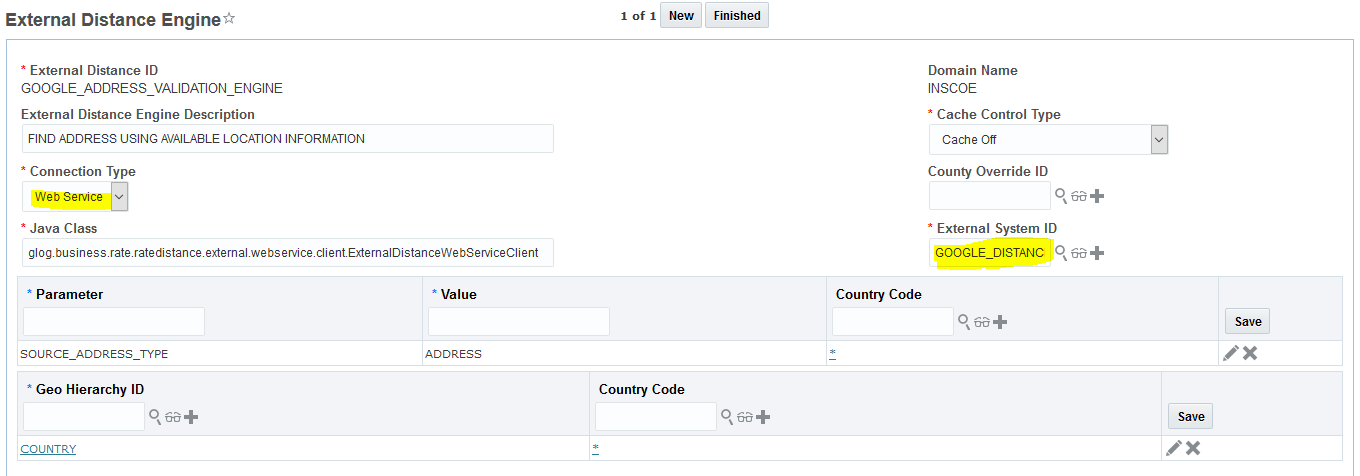
## Web Service

Steps to configure (This is valid for both OTM cloud and On-premise versions)

* Create Web service in OTM by configuring your custom application WSDL file. It will look like below.
* Create External system to configure above web service.



* Finally, use this Web Service within External Distance Engine setup.

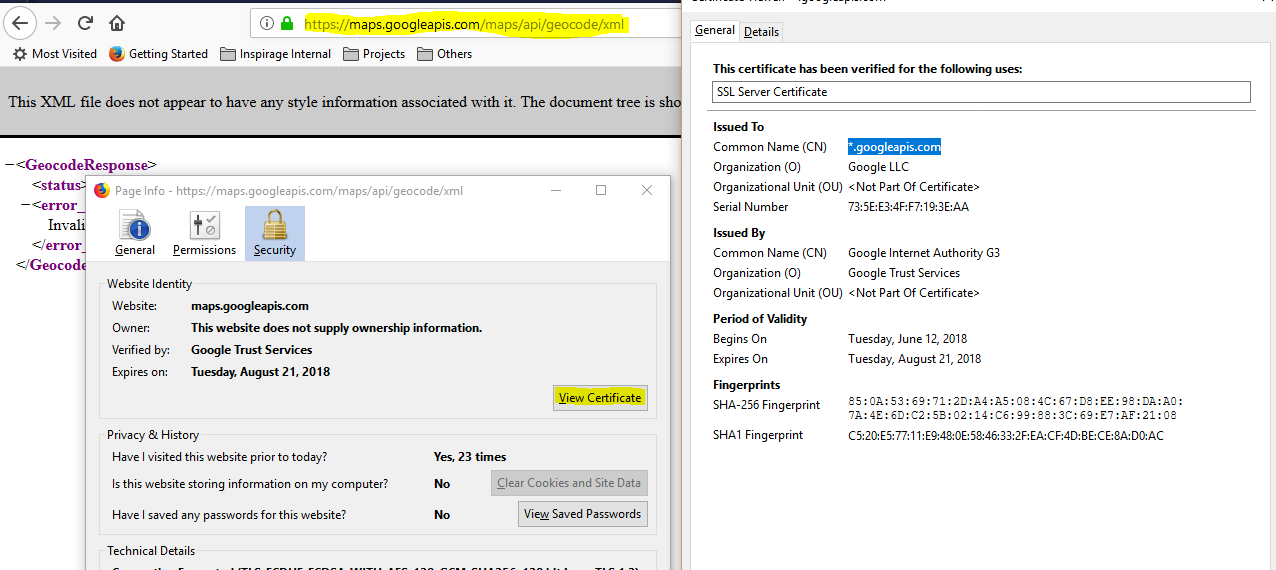


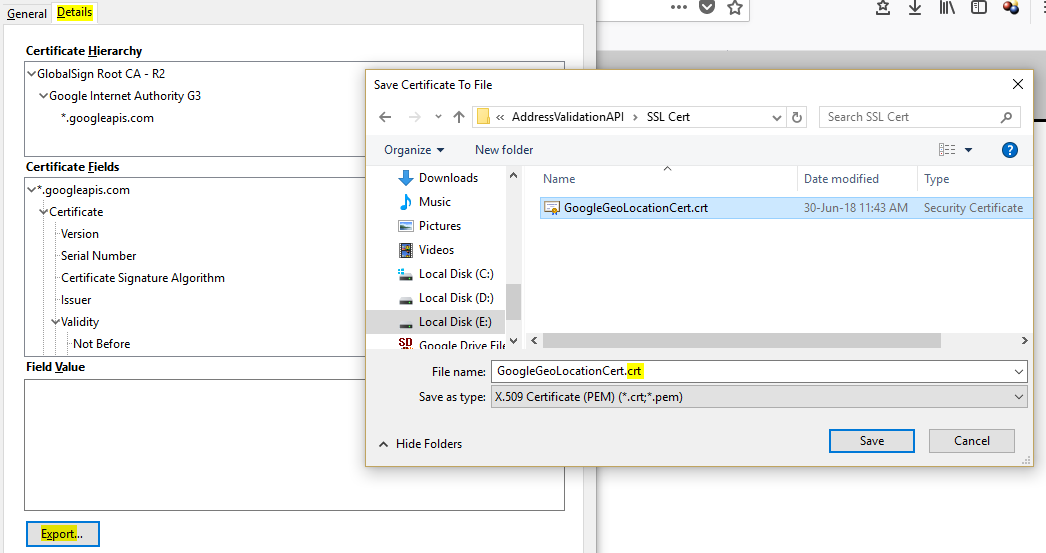
* We need to ensure that, service endpoint of Web service application should be accessible from OTM server.

# SSL Configuration (Only for on-premise)

This step is required to deploy Google API SSL certificates in OTM application server.

* Download the Google API certificate by accessing the link <https://maps.googleapis.com/maps/api/geocode/xml>
* Certificate can be downloaded or exported using any browser (Firefox, IE, Chrome and Edge etc.).
* Export this certificate as \*.crt file on your local machine. Screen shot below.





* After downloading this certificate, we need to import into OTM JDK keystore. Following command can be used.

**Command**:

keytool -import -trustcacerts -keystore $GLOG\_HOME/otm/jdk/jre/lib/security/cacerts -storepass changeit -noprompt -alias google -file $GLOG\_HOME/otm/temp/GoogleGeoLocationCert.crt

* After certificate import is complete, we need to setup following properties in OTM server. These properties are related to Tomcat and Weblogic.

**Changes in tomcat.conf and weblogic.conf**

jvm.arg=-Dweblogic.ssl.JSSEEnabled=true

jvm.arg=-Djava.protocol.handler.pkgs=com.sun.net.ssl.internal.www.protocol

jvm.arg=-Dssl.SocketFactory.provider=com.sun.net.ssl.internal.SSLSocketFactoryImpl

jvm.arg=-DUseSunHttpHandler=true

* Finally, restart OTM application to take the effect of newly added certificate in OTM keystore.

# Google GeoLocation API Key Generation

Follow these steps to get an API key:

1. Go to the Google Cloud Platform Console.

https://console.developers.google.com/flows/enableapi?apiid=geolocation&reusekey=true

1. Create or select a project.
2. Click **Continue** to enable the API.
3. On the **Credentials** page, get an **API key**.   
   Note: If you have an existing unrestricted API key, or a key with server restrictions, you may use that key.
4. From the dialog displaying the API key, select **Restrict key** to set a server restriction on the API key.
5. In the **Key restriction** section, select **IP addresses (web servers, cron jobs, etc.)**, follow the on-screen instructions to add server IP addresses, then click **Save**. Read more about [restricting API keys](https://developers.google.com/maps/documentation/geolocation/get-api-key#key-restrictions).
6. (Optional) Enable billing. See [Usage Limits](https://developers.google.com/maps/documentation/geolocation/usage-limits) for more information.

# Test Case

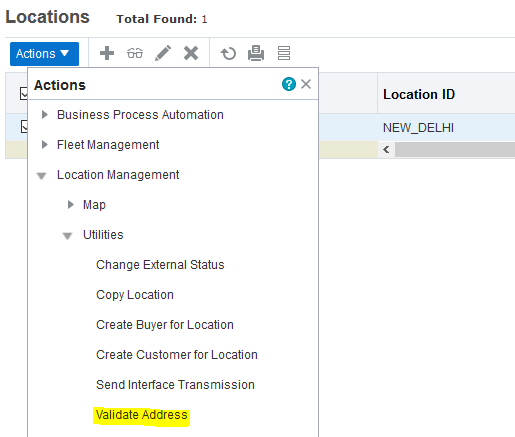
After completion of all the above steps, we need to perform a sample Test case to ensure all the configuration and deployment working properly.

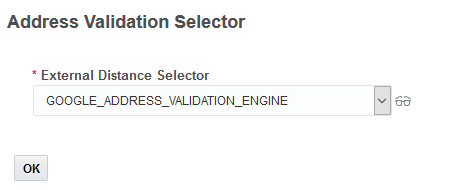
Steps:

* Create one Test Location in OTM using below navigation.

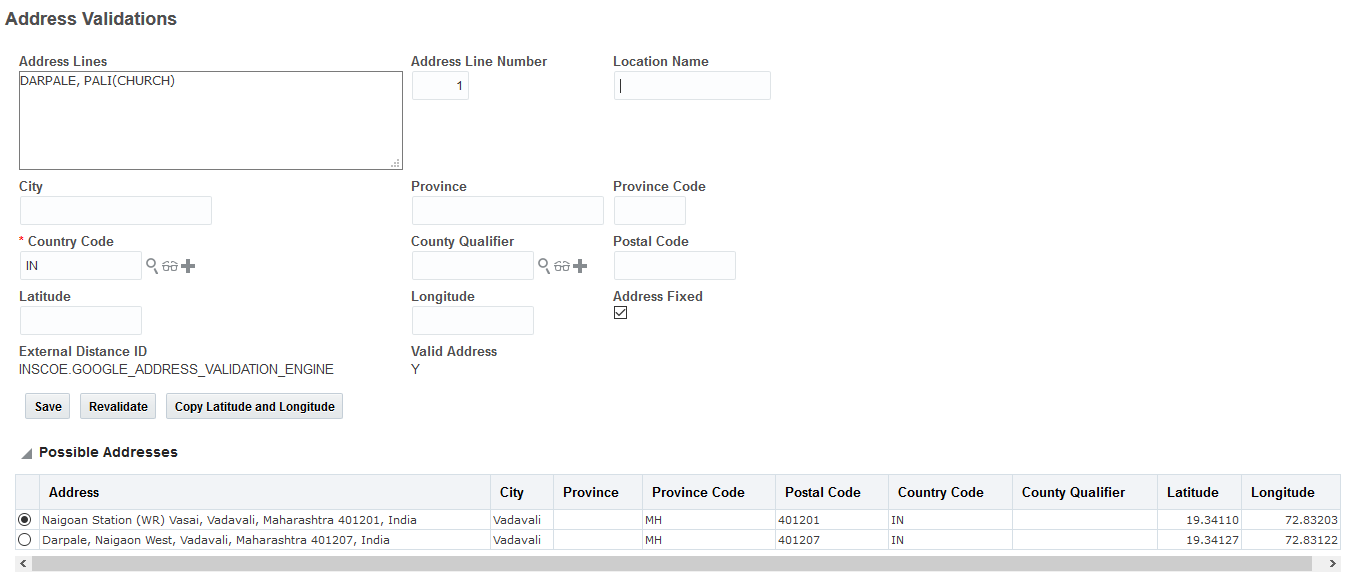
Shipment Management > Location Manager

* Once Location is created, select Validate Address action on Location object and select your Distance Engine ID.





* Click OK to see the result integrated from Google service as shown below.



* Above result confirm the Test case as successful.
* Similar steps can be performed when connection type in EDE configuration is Web Service.

# Source Code

Source code is available in below Google Drive Link.

<https://drive.google.com/drive/folders/1ZpMr0R3U_BUoA7bKXu06pgeAj0BBwZWg?usp=sharing>

# Open & Closed Issues

| # | Issue Description | Issue Date | Status | Responsibility | Resolution Date |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| 1 | Web service-based application implementation on Tomcat 7+ version | 28-Jun-2018 | Medium | Akshay Thakur | 02-July-2018 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |