

# Jio Developer guide Partner APIs protected with Mutual SSL & Oauth2.0

**COPYRIGHT INFORMATION**

**No part of this document may be copied, reproduced, stored in any retrieval system, or transmitted in any form or by any means, either electronically, mechanically, or otherwise without prior written permission.**

 Copyright Reliance Jio Infocomm Ltd.

**All rights reserved.**

Reliance Jio Infocomm Ltd., Reliance Corporate Park

# **Mutual SSL**

## 4.1.1 Introduction

This chapter identifies the document and the engagement to which it relates, describes the contents of the document, and states its purpose.

### Purpose

This document provides a guide to partner’s developers who would like to consume Jio APIs from their application server. The document is primarily meant for

* Outlining key steps to be followed
* Failure responses and reasons for the same

### Intended Audience for this Document

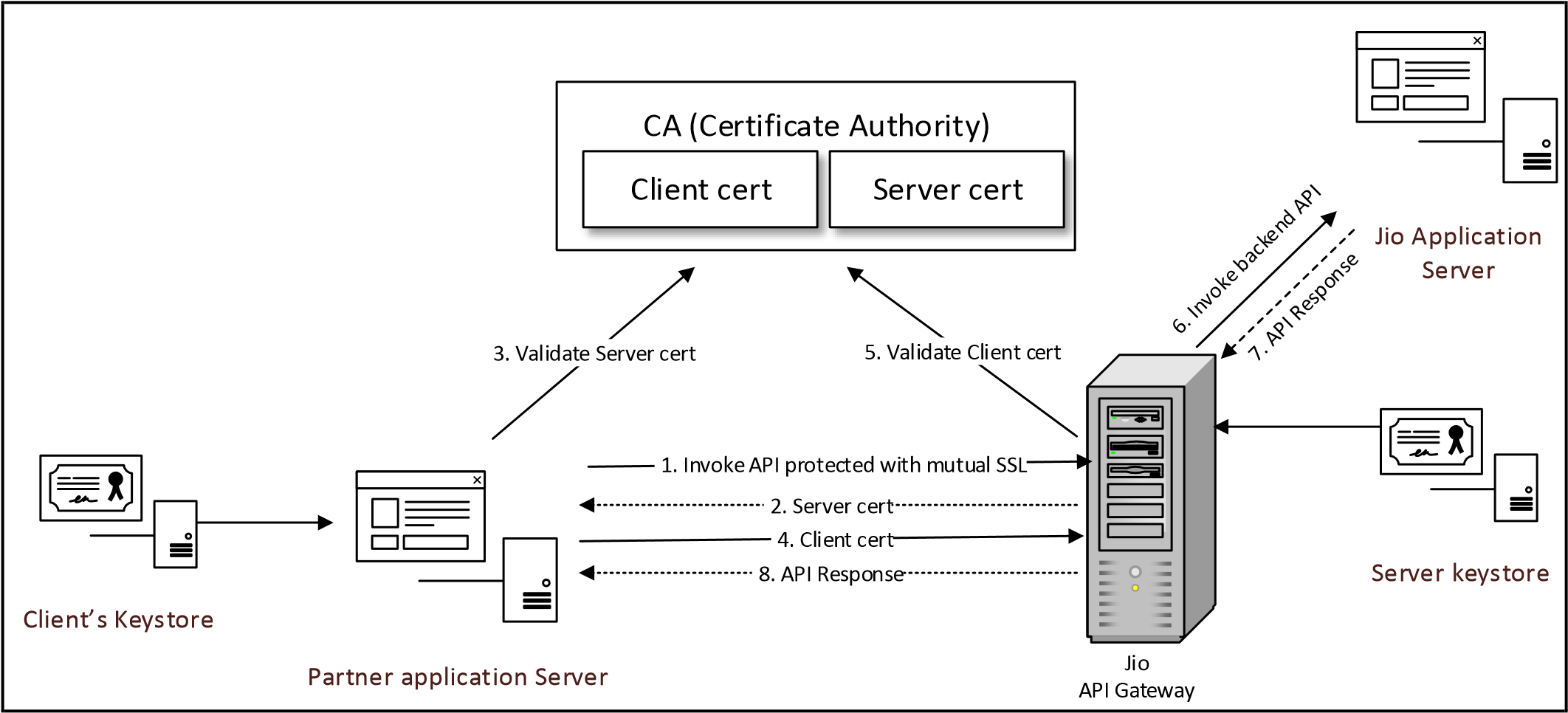
This document provides necessary details to Jio’s partner’s developers and architects on consumption of Jio APIs that are recommended to be consumed via mutual SSL.

### API Gateway details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Extrenal Partner Environment** | **Jio Environment** | **API Gateway hostname** |  | **Port\*** |
| Integration | SIT | api-sit.jio.com | 7443 |  |
| Pre-prod | Replica | api.jiolabs.com | 7443 |  |
| Production | Production | api.ril.com | 7443 |  |

**\*** APIs that are not protected via mutual SSL, https default port # 443 should be used

### Mutual SSL or 2way SSL



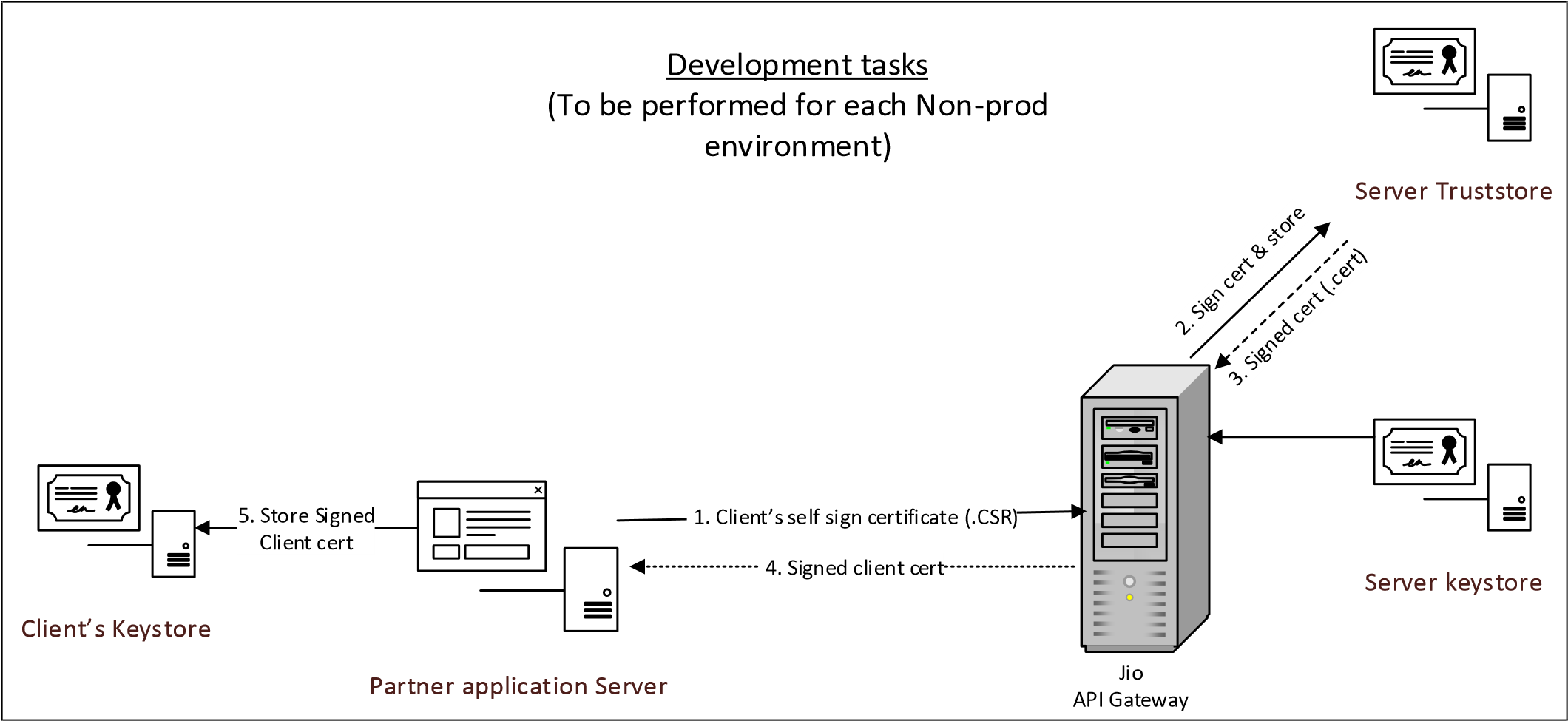
In case of two-way SSL or mutual SSL, both client and server authenticate each other to ensure that both parties involved in the communication are trusted. Both parties share their public certificates to each other and then verification/validation is performed based on that.

Below is the description of the steps involved in establishment of connection and transfer of data between a client and server in case of mutual SSL -

1. Client requests a protected resource over HTTPS protocol and the SSL/TSL handshake process begins.
2. Server returns its public certificate to the client along with server hello.
3. Client validates/verifies the received certificate. Client verifies the certificate through certification authority (CA) for CA signed certificates.
4. If Server certificate was validated sucessfully, client will provide its public certificate to the server.
5. Server validates/verifies the received certificate through certification authority (CA) for CA signed certificates.
6. After completion of handshake process, client and server communicate and transfer data with each other encrypted with the secret keys shared between the two during handshake.

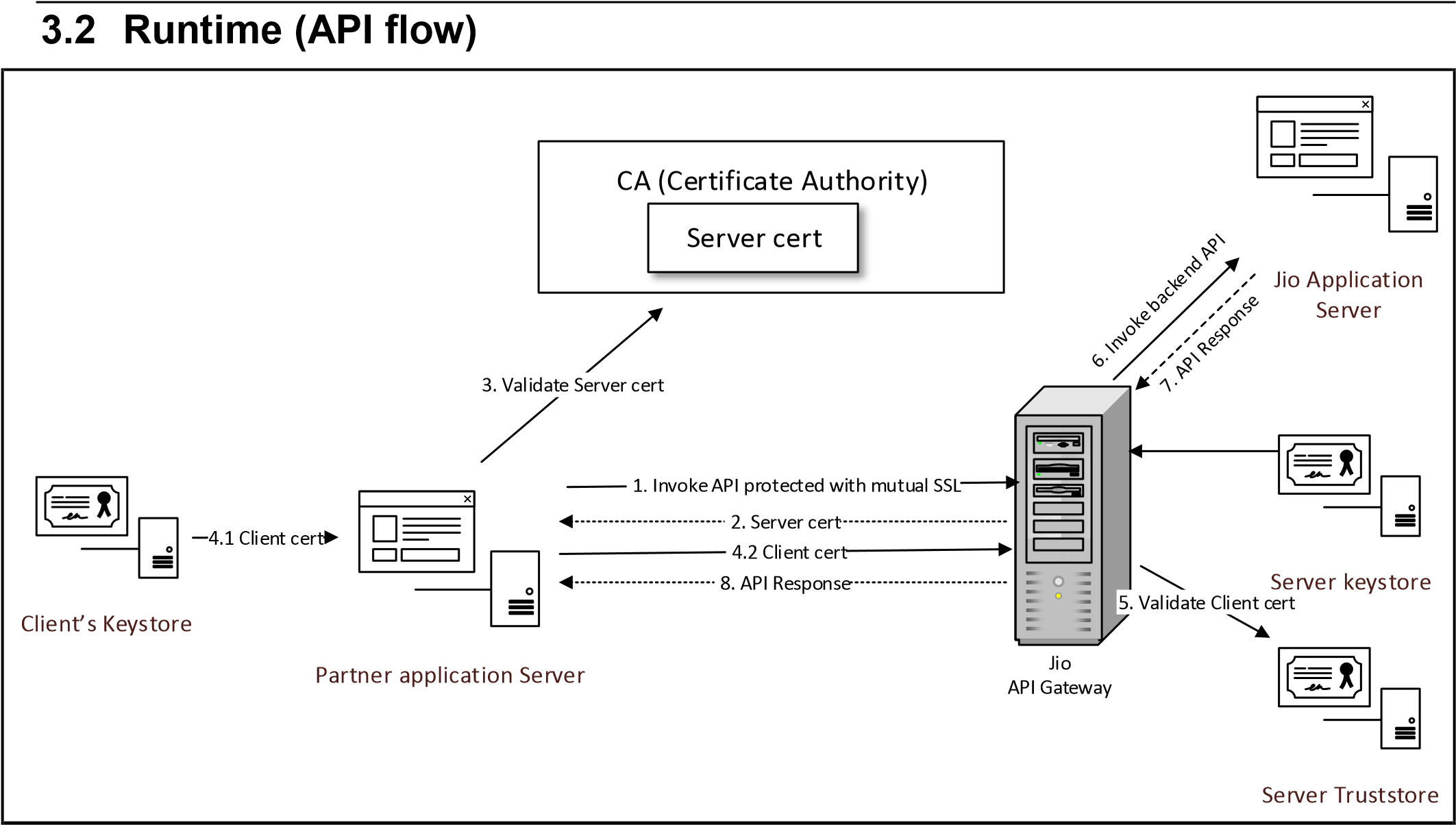
### Setup mutual SSL

### Design time (pre-requisites)



CA signed certificates are not mandatory for Jio’s Non-Prod environment and to support self-signed certificates, following steps should be followed prior to invocation of APIs protected via mutual SSL -

1. Partner developer shares the .CSR file for the self-signed certificate.
2. Server signs the the certificate and store it in its trust store.
3. Server trust store returns the signed certificate (.cert file) to Jio API gateway.
4. API gateway team will share the .cert file with the partner development team
5. Partner development team needs to share this certificate in their keystore



1. Client requests a protected resource over HTTPS protocol and the SSL/TSL handshake process begins.
2. Server returns its public certificate to the client along with server hello.
3. Client validates/verifies the received certificate. Client verifies the certificate through certification authority (CA) for CA signed certificates.
4. If Server certificate was validated sucessfully, client will provide its public certificate to the server.
5. Server validates/verifies the received certificate through it is own trust store.
6. After completion of handshake process, client and server communicate and transfer data with each other encrypted with the secret keys shared between the two during handshake.

### Sample Java code (Client side)

|  |
| --- |
| import java.net.URL; import java.io.\*; import javax.net.ssl.HttpsURLConnection; public class TestHttps  {  private static final String CA\_KEYSTORE\_PATH = "C:\\Program Files\\Java\\jdk1.8.0\_45\\jre\\lib\\security\\cacerts";  private static final String CA\_KEYSTORE\_PASS= "changeit"; private static final String CLIENT\_KEYSTORE\_PATH = "D:\\Files\\Certs\\keystore.jks"; private static final String CLIENT\_KEYSTORE\_PASS = "password";  public static void main(String[] args) throws Exception  {  System.setProperty("javax.net.ssl.trustStore",CA\_KEYSTORE\_PATH);  System.setProperty("javax.net.ssl.trustStorePassword",CA\_KEYSTORE\_PASS);  System.setProperty("javax.net.ssl.keyStore",CLIENT\_KEYSTORE\_PATH);  System.setProperty("javax.net.ssl.keyStorePassword",CLIENT\_KEYSTORE\_PASS);  String httpsURL = "https://cluster\_host:8443/testcert";  URL myurl = new URL(httpsURL);  HttpsURLConnection con = (HttpsURLConnection)myurl.openConnection();  InputStream ins = con.getInputStream();  InputStreamReader isr = new InputStreamReader(ins);  BufferedReader in = new BufferedReader(isr); String inputLine; while ((inputLine = in.readLine()) != null)  {  System.out.println(inputLine);  } |

### FAQs

1. Error observed on client side - javax.net.ssl.SSLHandshakeException

*javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException: PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target*

Server side analysis – Either, the client is sending wrong certificate or no certificate with the request.

*com.l7tech.server.policy.assertion.ServerAuditDetailAssertion: 151: Variable*

*'request.ssl.clientCertificate' is not a String and cannot be converted to one. (Actual type: sun.security.x509.X509CertImpl)*

*com.l7tech.server.policy.assertion.identity.ServerMemberOfGroup: 4208: Authentication failed for TPA*

*com.l7tech.server.policy.assertion.ServerAuditDetailAssertion: -4: Policy Fragment: SECO-*

*ClientCertValidationError-7049*

## OAuth2.0 for OID

OAuth 2.0 framework provides authentication and authorization.

**Steps to be followed for accessing API’s secured by OAuth**

### Generate Access Token

Accessing an **OAuth** protected API needs to be sent with **access\_token** along with other required parameters for it. In order to get the **access\_token** user needs to be authenticated first by sending the user credentials against user store (**OID** or **IDAM** or **JIOAD** or **RR mHere** or **SCRUM and PRMS**)

Access the below **URL** with required query and header parameters for **getting/generating** the **access\_token**.

**URL:**

https://<**SECO API Domain** or **IP Address**>:<Port>/auth/oauth/v2/token?**state**=state\_test&**scope**=<scope>&**client\_id**=<**api key**>&client\_secret=<**client secret**>&**grant\_type**=password

**Query Parameters:**

Change the below specified query parameter values only which are obtained from the **SECO API Portal** and keeping the remaining as it is.

**client\_id:** API Key which is obtained from SECO after registering an Application/App

**client\_secret:** API Secret which is obtained from SECO after registering an Application/APP

**scope:** scope which is obtained from SECO after registering an Application/APP.

scope = agent -- For OID

**Header Parameters:**

Sending login/authentication credentials in the request header

**username: <**username**> (**user name existing in OID/IDAM/JIOAD/ADLDS/HNH/RR mHere/SCRUM and PRMS) -- Required

**password: <**password> (Password of the user) – Required

**ttl:<**no. of seconds**> (**No. of seconds the oauth token should/will be active**) --** Optional

**Note:**  Header parameter **ttl** should be an integer value and default value is **3600 seconds**

**Http Method:**  POST

**Success Response (for scope=agent):**

**{**

**"access\_token": "303f564a-2fd9-4199-9f2b-76f7f68f3bd7",**

**"token\_type": "Bearer",**

**"expires\_in": "3600",**

**"refresh\_token": "32ba2b66-ad23-4541-b7a4-4915ef79e733",**

**"scope": "agent",**

**"associate": {**

**"agentCode": "youragentcode",**

**"aadhaarId": "",**

**"name": "yourname",**

**"Role": "",**

**"areaOffice": "",**

**"functionalManagerId": "",**

**"lineManagerId": "",**

**"parentCompany": "",**

**"Organization": {**

**"RJILDepartment": "",**

**"Org": "",**

**"name": "",**

**"distributionCenter": "",**

**"city": "",**

**"state": "",**

**"circleId": ""**

**},**

**"ContactDetails": {**

**"email": "",**

**"phoneNumber": ""**

**},**

**"memberOfGroups": [],**

**"group": ""**

**}**

**}**

**Note:**

* **agentCode** value needs to be set as **uid** value in subsequent calls.
* **uid** will be used in accessing  the **API’s** and **refresh\_token** service after successful authentication along with other required parameters.
* Also the consumers should store the **refresh\_token** and other user related information on the **client side** so that those can be used whenever required in subsequent calls.

**Possible Failure Responses:**

**1)**

**{**

**"success": false,**

**"errors": [**

**{**

**"code": “400”,**

**"reason": "invalid\_grant",**

**"details": "Resource Owner authentication failed"**

**}**

**]**

**}**

**2.**

**{**

**"success": false,**

**"errors": [**

**{**

**"code": "RIL4G\_T\_SECO\_7026",**

**"reason": "OAuth access token life time is not valid",**

**"details": "OAuth access token life time should be a number"**

**}**

**]**

**}**

**3.**

**{**

**"success": false,**

**"errors": [**

**{**

**"code": "RIL4G\_T\_SECO\_7027",**

**"reason": "OAuth access token life time is not valid",**

**"details": "Max. allowed time for access token is 86400sec"**

**}**

**]**

**}**

### Accessing OAuth Protected API

After successful authentication and obtaining the access\_token call the required API that’s protected with OAuth with required parameters and **access\_token** in authorization header.

**URL:**

https://<**SECO API Domain or IP Address**>:<Port>/<URI API>

**Http Method:** POST/GET

**Header Parameters:**

**Authorization:** Bearer **<**access token value**> (Access token** is one of value from response obtained after successful authentication in step 1**)**

**uid: <**agentcode> **(agentCode** is a value from response obtained after successful authentication in step 1**)**

**Success Response:**

**{**

“success”: true,

                <Other data based of the API>

**}**

**Failure Response:**

If the failure is due to invalid or expired Access token, following error is returned.

**{**

**"success": false,**

**"errors": [**

**{**

**"code": “**0**”,**

**"reason": "invalid\_request",**

**"details": "Validation error"**

**}**

**]**

**}**

### Refresh Token

Refresh token is request is to be used when the **access token** is expired and want to get the new **access token** without sending the user credentials

**URL:**

https:// <**SECO Domain or IP Address**>:<Port>/auth/oauth/v2/token?client\_id=<**api key**>&client\_secret=<**client secret**> &grant\_type=refresh\_token&refresh\_token=<**refresh token**>&uid=<userid/agentcode>

**Http Method:** POST

**Query Parameters:**

Set values for these query parameters which are obtained from the SECO API Portal and successful authentication

**client\_id:** API Key which is obtained from SECO after registering an Application/App

**client\_secret:** API Secret which is obtained from SECO after registering an Application/APP

**refresh\_token:** Refresh token is obtained in the response from SECO after successful authentication

**Success Response:**

**{**

 **"access\_token": "c51d1431-060a-43c3-b411-96fd508b5e05",**

    "token\_type": "Bearer",

    "expires\_in": 3600,

**"refresh\_token": "8aa5efe3-b1fc-4c9f-8b6c-116061bcecb6"**,

    "scope": "agent",

    "uid": "yourid"

**}**

**Failure Response:**

If the failure is due to invalid Refresh token being sent, following error is returned.

**{**

**"success": false,**

**"errors": [**

**{**

**"code": “RIL4G\_T\_SECO\_7021”,**

**"reason": "invalid\_request",**

**"details": "Invalid Request"**

**}**

**]**

**}**

### Validate OAuth access\_token

This service is used to verify whether the **access\_token** user has is valid one or not. Any consumer/application having **access\_token** and wanted to interact with other systems without **SECO** should be using this **SECO** service to valid the requestor (i.e. consumer/application) and then only allow or deny the access accordingly.

**URL:**

https://<**SECO API Domain or IP Address**>:<Port>/validate/oauth/token

**Http Method:** POST

**Header Parameters:**

**Authorization:** Bearer **<**access token value**> (Access token** is one of value from response obtained after successful authentication in step 1**)**

**uid: <**agentcode> **(agentCode** is a value from response obtained after successful authentication in step 1**)**

**X-API-Key :** API key obtained from API portal (after adding this API to your application)

**Success Response:** In success response **Http Response Status Code** will be **200 OK** with no content or data.

**Failure Response:**

If the failure is due to invalid or expired Access token, following error is returned.

**{**

**"success": false,**

**"errors": [**

**{**

**"code": “0”,**

**"reason": "invalid\_request",**

**"details": "Validation error"**

**}**

**]**

**}**

### Revoke Token

Revoke token request is to be used when the **access token** generated using [Generate Access Token](#_Generate_Access_Token) or [Refresh Token](#_Refresh_Token) need to be deleted from OAuth server. This is usually done when user wants to logout from the OAuth application.

**URL:**

https://<**SECO Domain or IP Address**>:<Port>/auth/oauth/v2/token/revoke?client\_id=<**api key**>&client\_secret=<**client secret**>&token=<accesstoken>&uid=<userid/agentcode> &token\_type\_hint=access\_token

**Http Method:** POST

**Query Parameters:**

Set values for these query parameters which are obtained from the SECO API Portal and successful authentication

**client\_id:** API Key which is obtained from SECO after registering an Application/App

**client\_secret:** API Secret which is obtained from SECO after registering an Application/APP

**token:** Access token obtained in the response after successful authentication from [step1](#_Generate_Access_Token)/[step3](#_Refresh_Token)

**uid: <**agentcode> obtained after successful authentication from step 1

**Success Response:** In success response **Http Response Status Code** will be **200 OK** with no content or data.

**Failure Responses:**

1. If the failure is due to invalid token being sent, following error is returned.

**{**

**"success": false,**

**"errors": [**

**{**

**"code": "RIL4G\_T\_SECO\_7021",**

**"reason": "invalid\_request",**

**"details": "Invalid Request"**

**}**

**]**

**}**

1. If mandatory parameters are not sent in the request.

**{**

**"success": false,**

**"errors": [**

**{**

**"code": "RIL4G\_T\_SECO\_7016",**

**"reason": "Bad Request",**

**"details": "Input data required is missing"**

**}**

**]**

**}**