

AQUILA - Salesforce Integration Using JWT Authentication

Salesforce requires secure communication protocols for authorization and data exchange between external applications and Salesforce orgs. This involves creating digital certificates, configuring external client apps, and establishing secure authentication methods. OpenSSL provides the cryptographic tools needed to generate private keys and self-signed certificates for secure communication over networks.

Integration Overview

This integration supports secure communication through:

- JWT (JSON Web Token) authentication using digital certificates
- OAuth authentication with external client apps
- Self-signed certificates and keystore management

Organizations can authorize Salesforce CLI commands and establish secure API connections using these authentication methods.

Compatibility

- Supports Salesforce CLI authorization via JWT Bearer Flow
- Compatible with macOS, Linux, and Windows operating systems
- Requires OpenSSL for certificate generation
- Requires Java keytool for keystore conversion (optional)

Installing OpenSSL in your Log Collector

OpenSSL is an open-source software library that provides tools and protocols for secure communication over networks. It helps encrypt data so that information like passwords, credit card numbers, and private messages stay secure when sent over the internet.

Step 1:

In Linux:

Install OpenSSL on your system:

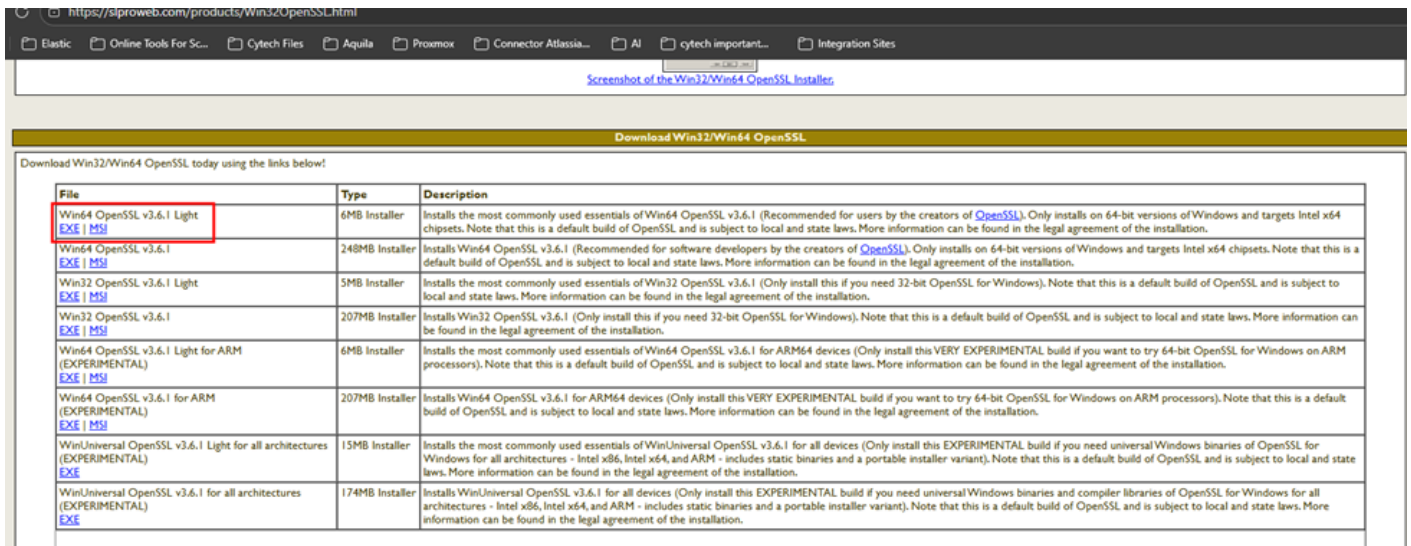
```
sudo apt install openssl
```

```
testing- :~$ sudo apt install openssl
```

In Windows:

Download the Installer:

1. Go to the [Shining Light Productions](#) page and download the "Win64 OpenSSL Light" EXE installer (current version 3.x is recommended).



Download Win32/Win64 OpenSSL today using the links below!

File	Type	Description
Win64 OpenSSL v3.6.1 Light EXE MSI	6MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v3.6.1 (Recommended for users by the creators of OpenSSL). Only installs on 64-bit versions of Windows and targets Intel x64 chipsets. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v3.6.1 EXE MSI	248MB Installer	Installs Win64 OpenSSL v3.6.1 (Recommended for software developers by the creators of OpenSSL). Only installs on 64-bit versions of Windows and targets Intel x64 chipsets. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v3.6.1 Light EXE MSI	5MB Installer	Installs the most commonly used essentials of Win32 OpenSSL v3.6.1 (Only install this if you need 32-bit OpenSSL for Windows). Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v3.6.1 EXE MSI	207MB Installer	Installs Win32 OpenSSL v3.6.1 (Only install this if you need 32-bit OpenSSL for Windows). Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v3.6.1 Light for ARM (EXPERIMENTAL) EXE MSI	6MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v3.6.1 for ARM64 devices (Only install this VERY EXPERIMENTAL build if you want to try 64-bit OpenSSL for Windows on ARM processors). Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v3.6.1 for ARM (EXPERIMENTAL) EXE MSI	207MB Installer	Installs Win64 OpenSSL v3.6.1 for ARM64 devices (Only install this VERY EXPERIMENTAL build if you want to try 64-bit OpenSSL for Windows on ARM processors). Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
WinUniversal OpenSSL v3.6.1 Light for all architectures (EXPERIMENTAL) EXE	15MB Installer	Installs the most commonly used essentials of WinUniversal OpenSSL v3.6.1 for all devices (Only install this EXPERIMENTAL build if you need universal Windows binaries of OpenSSL for Windows for all architectures - Intel x86, Intel x64, and ARM - includes static binaries and a portable installer variant). Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
WinUniversal OpenSSL v3.6.1 for all architectures (EXPERIMENTAL) EXE	174MB Installer	Installs WinUniversal OpenSSL v3.6.1 for all devices (Only install this EXPERIMENTAL build if you need universal Windows binaries and compiler libraries of OpenSSL for Windows for all architectures - Intel x86, Intel x64, and ARM - includes static binaries and a portable installer variant). Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.

2. Run the Installer: Execute the downloaded file and follow the on-screen instructions.

3. Installation Path: It is generally recommended to install in the default location, typically C:\Program Files\OpenSSL-Win64.

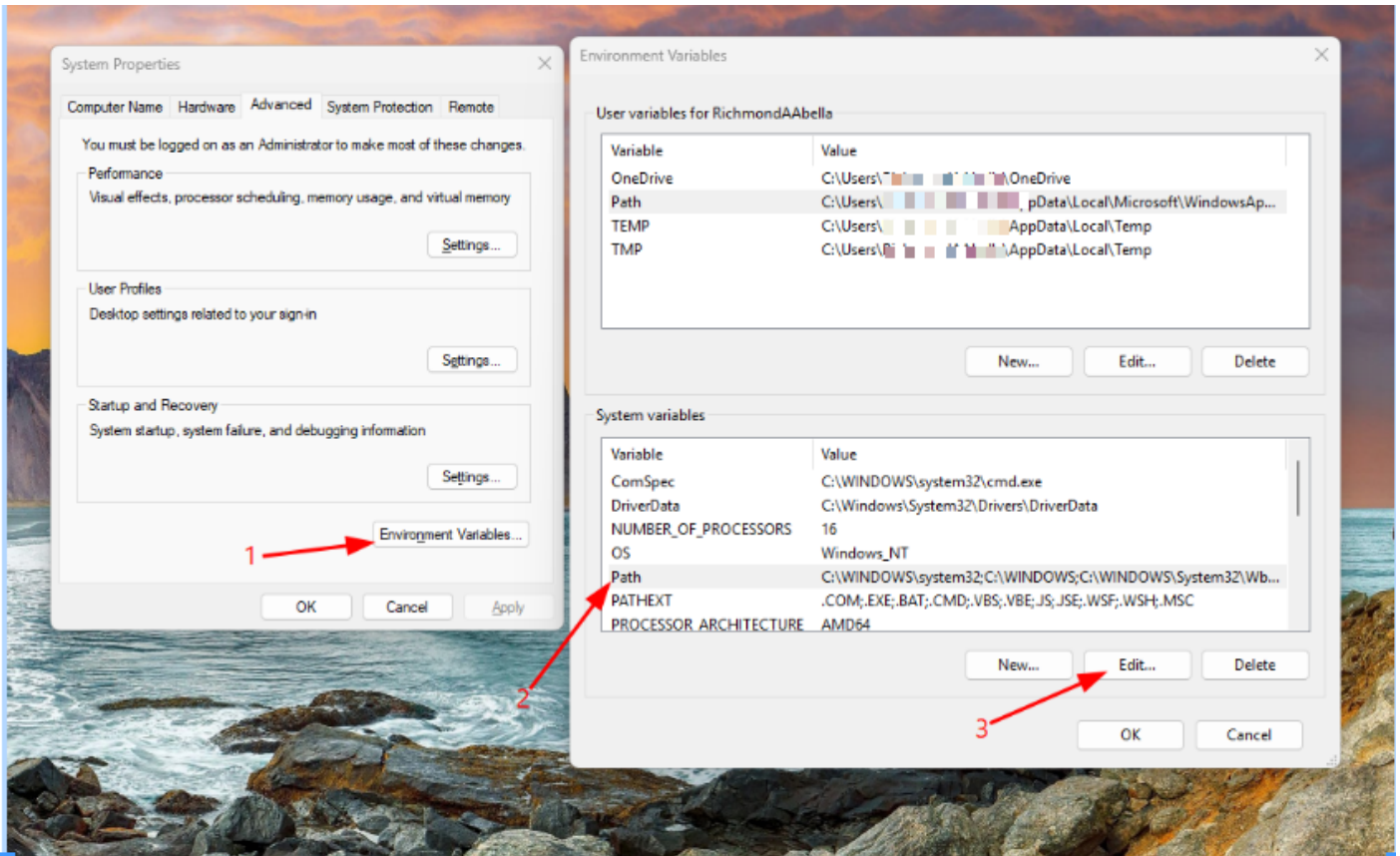
4. Configure System Environment Variable:

Search for "Environment Variables" in the Windows search bar.

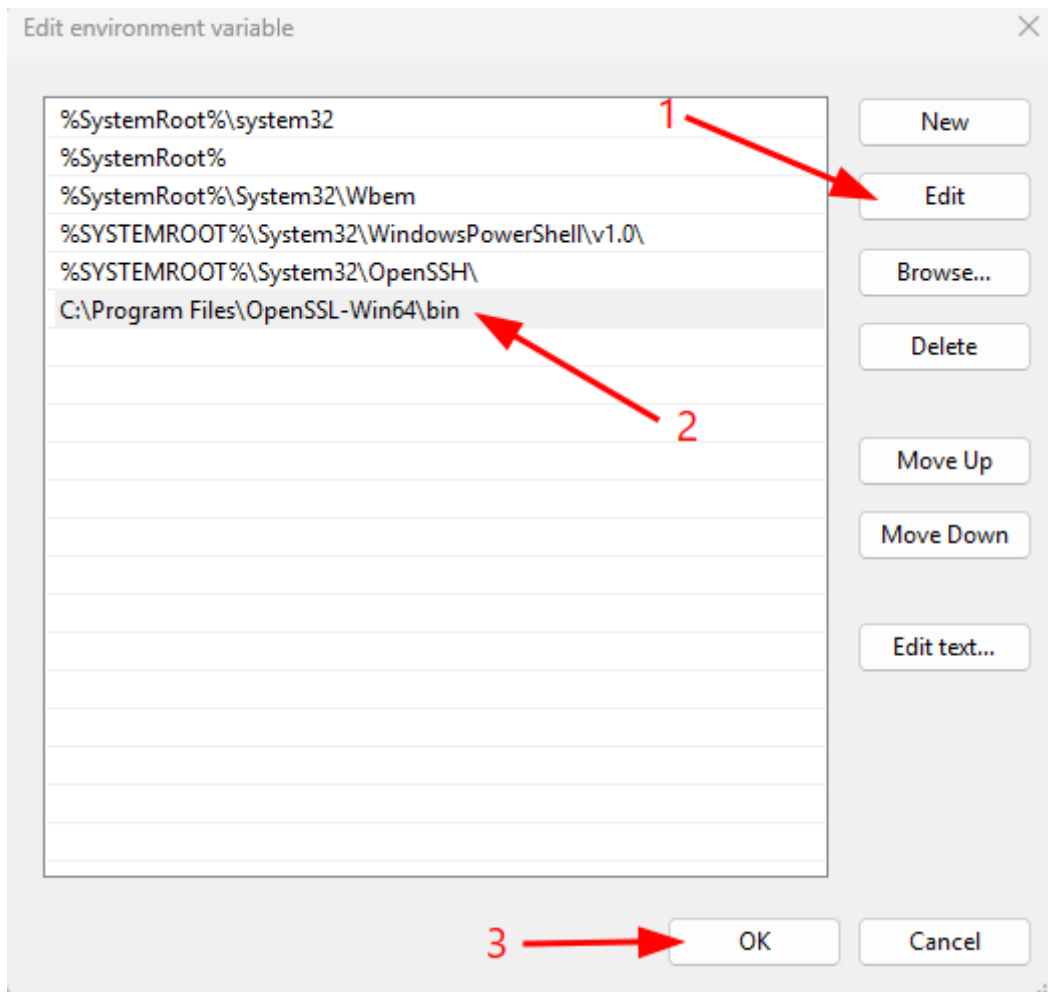
Click "Edit the system environment variables".

Click "Environment Variables".

Under "System Variables", find and select Path, then click "Edit".



Click "Edit" and add the path to the bin folder, for example: C:\Program Files\OpenSSL-Win64\bin. Click OK on all windows to save.



Step 2:

Verify OpenSSL installation by running:

- macOS/Linux: `which openssl`
- Windows: `where openssl`

```
testing@testing:~$ which openssl
/usr/bin/openssl
```

Creating a Private Key and Self-Signed Digital Certificate

A digital certificate and the private key used to sign the certificate are needed to authorize an organization using the `org login jwt` command. While it is strongly advised to utilize a certificate issued by a certifying authority, you can use OpenSSL to generate a self-signed certificate to get started.

This process produces two files:

- **server.key** — The private key used when authorizing an org with the `org login jwt` command
- **server.crt** — The digital certificate uploaded when creating the required external client app

Step 1:

Open a terminal (macOS and Linux) or command prompt (Windows).

Step 2:

Create a directory to hold the generated files and navigate to it:

```
mkdir /Users/jdoe/JWT
cd /Users/jdoe/JWT
```

Step 3:

Create a private key and save it as `server.key` file:

Remember to change "**<your password>**" to the password of your choice. The password should be the same with the **server.pass.key** and **server.key**.

- `server.pass.key` command

```
openssl genpkey -aes-256-cbc -algorithm RSA -pass pass:<your password> -out server.pass.key -pkeyopt rsa_keygen_bits:2048
```

```
testing-@:~$ openssl genpkey -aes-256-cbc -algorithm RSA -pass pass:1234567890 -out server.pass.key -pkeyopt rsa_keygen_bits:2048
```

- server.key command

```
openssl rsa -passin pass:<your password> -in server.pass.key -out server.key
```

```
testing-@:~$ openssl rsa -passin pass:password -in server.pass.key -out server.key
```

Step 4:

Use the server.key file to create a certificate signing request and save it as server.csr:

When prompted, provide your organization's details. Enter only the **Country Name, State or Province, Locality, and Organization Name**—you may leave all other fields blank.

Do not enter a password when generating the `server.csr`, as it may cause an authentication mismatch.

```
openssl req -new -key server.key -out server.csr
```

```
testing-@:~$ openssl req -new -key server.key -out server.csr
```

Step 5:

Create a self-signed digital certificate using the server.key and server.csr files:

```
openssl x509 -req -sha256 -days 365 -in server.csr -signkey server.key -out server.crt
```

```
testing-@:~$ openssl x509 -req -sha256 -days 365 -in server.csr -signkey server.key -out server.crt
```

Step 6:

Clone the server.key file and save it as server.pem

Important step to successfully integrate into SIEM

```
cp server.key server.pem
```

Creating an External Client App in Your Salesforce Organization

Salesforce CLI requires an external client app in the org that you're authorizing. An external client app is a packageable framework that enables a third-party application (Salesforce CLI) to integrate with Salesforce using APIs and security protocols. You must create your own external client app when authorizing the org with the `org login jwt` command.

Step 1:

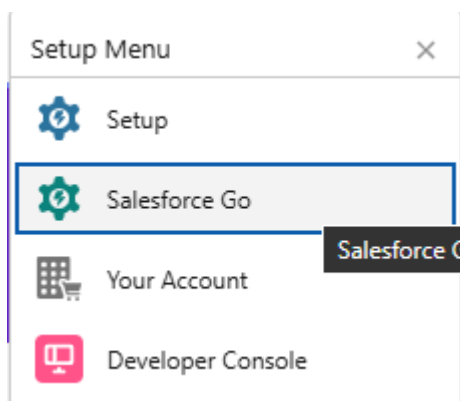
Log in to your Salesforce Organization.

Note: If the salesforce dashboard interface is in classic mode change it to lightning mode.

- In the Upper Right Corner click the gear icon.



- Select salesforce go.



Step 2:

From the Quick Find box in Setup, enter **App Manager**, then click **App Manager**.

Search Setup

Setup Home Object Manager

app manag

Lightning Experience App Manager

New Lightning App New External Client App

33 items • Sorted by App Name • Filtered by All appmenutems - TabSet Type, App Type

App Name ↑	Developer Name	Description	Last Modified ...	App Type	Vi...
1	AllTabSet		6/30/2025, 12:16 A...	Classic	
2		ool T...	7/1/2025, 5:44 AM	Connected (Managed)	
3		A	6/30/2025, 12:16 A...	Classic	
4		M	6/30/2025, 12:16 A...	Lightning	✓
5		A	6/30/2025, 12:18 A...	Lightning	✓
6		E	6/30/2025, 12:16 A...	Lightning	✓
7		S	6/30/2025, 12:16 A...	Classic	
8		S	7/2/2025, 10:52 AM	Classic	
9		ip	6/30/2025, 12:16 A...	Connected	
10		T	7/1/2025, 5:43 AM	Connected (Managed)	
11		T	7/1/2025, 5:43 AM	Connected (Managed)	
12		N	6/30/2025, 12:16 A...	Lightning	✓
13		T	7/1/2025, 5:44 AM	Connected (Managed)	

Step 3:

Click **New External Client App**.

Step 4:

Update the basic information as needed, such as the external client app name and your contact email address.

Note: The email address provided must be valid, as **Salesforce** will use it to communicate with your team regarding any updates or issues related to your application usage.

Basic Information

* External Client App Name my external app	* API Name my_external_app
* Contact Email [Redacted]	* Distribution State Local
Contact Phone Enter a phone number...	Info URL Enter a URL...
Logo Image URL Enter a URL... Choose one of our sample logos.	Icon URL Enter a URL... Choose one of our sample logos.
Description [Empty]	

> API (Enable OAuth Settings)

Cancel Create

Step 5:

Under **API (Enable OAuth Settings)**, click **Enable OAuth**.

API (Enable OAuth Settings)

Enable OAuth

Step 6:

Under **App Settings**, in the **Callback URL** box, enter the URL below:

https://<base-url>/callback

✓ API (Enable OAuth Settings)

Enable OAuth

App Settings

* Callback URL

Enter a URL...

To find the base URL and instance URL follow the guide below.

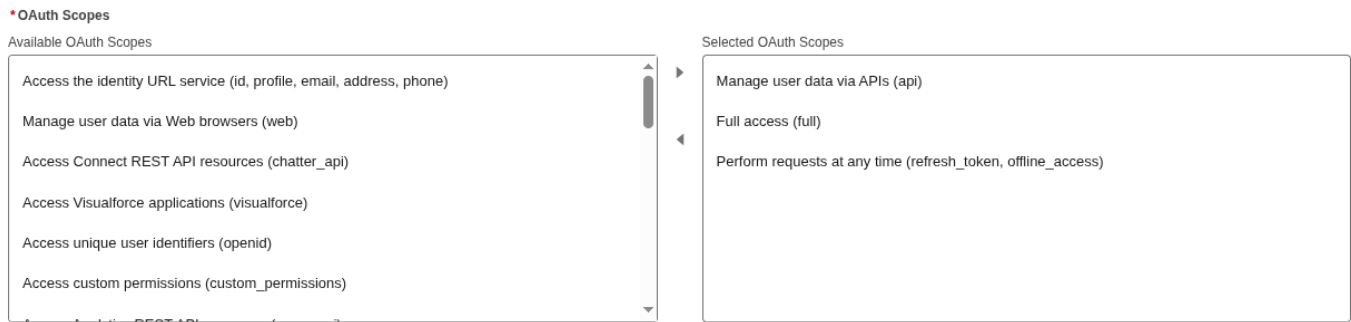
- In quick find box, enter **my domain** then select my domain under Company Settings.
- Under My Domain Details copy **Current My Domain URL** that's your base URL and Instance URL.

The screenshot shows the Salesforce 'My Domain' settings page. A search bar at the top left contains 'my domain'. Below it, the 'Company Settings' menu is expanded, and 'My Domain' is selected. The main content area is titled 'My Domain Settings' and includes a description: 'My Domain showcases your company's brand and keeps your data more secure. The domains that Salesforce hosts for your org include your company-specific My Domain name.' Under the 'My Domain Details' section, the 'Current My Domain URL' is highlighted with a red box and labeled '3'. The 'My Domain Name' is 'orgfarm-08c92d1946-dev-ed' and the 'Domain Suffix' is 'Standard (*.my.salesforce.com)'. The 'Routing and Policies' section shows 'Salesforce Edge Network' is enabled and the 'Routing Method' is set to 'Global'.

Step 7:

In the **OAuth Scopes** section, select these scopes:

- **Manage user data via APIs (api)** - Gives you access to user data.
- **Perform requests at any time (refresh_token, offline_access)** - Permits you to get an OAuth access token.
- **Full access (full)** - grant all access to the permission for integration.



Step 8:

(Required for JWT) In the **Flow Enablement** section, select **Enable Client Credentials Flow** and **Enable JWT Bearer Flow**.

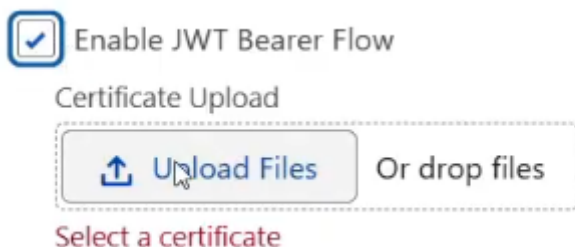
- **Enable Client Credentials Flow** - Allows your app to exchange its client credentials for an access token. And be able to access the credential Client ID.
- **Enable JWT Bearer Flow** - A secure, server-to-server authentication method used to integrate external applications with Salesforce without requiring manual user login.

Flow Enablement

- Enable Client Credentials Flow
- Enable Authorization Code and Credentials Flow
- Enable Device Flow
- Enable JWT Bearer Flow
- Enable Token Exchange Flow

Step 9:

(Required for JWT) Click **Upload Files** and upload your digital certificate file (**server.crt**).



Step 10:

In **Security** section check the following:

- Require secret for Web Server Flow
- Require secret for Refresh Token Flow
- Enable Refresh Token Rotation

Security

- Require secret for Web Server Flow
- Require secret for Refresh Token Flow
- Require Proof Key for Code Exchange (PKCE) extension for Supported Authorization Flows
- Enable Refresh Token Rotation
- Issue JSON Web Token (JWT)-based access tokens for named users

Step 11:

Click **Create** and **Edit** to configure additional settings.

Step 12:

(Required for JWT) Click the **Policies** tab and configure the following:

- Open **OAuth(Open Authorization) Policies**
- In the **Plugin Policies** section, set **Permitted Users** to **Admin approved users are pre-authorized**
- In **OAuth Start URL** use your organization base URL example: (https://fun-dream-996.my.salesforce.com/)
- Click **OK**
- In the **App Policies** section, select the profiles and permission sets that are pre-authorized to use this external client app

Plugin Policies

* Permitted Users

OAuth Start URL

Enter a URL...

All users can self-authorize

All users can self-authorize

Admin approved users are pre-authorized

Apex Plugin Class

Search classes

ed. You can define them [here](#) before assigning to your app.

App Policies

* Start Page ⓘ

None

Select Profiles

Available Profiles

- Analytics Cloud Integration User
- Chatter Free User
- Analytics Cloud Security User

Selected Profiles

- System Administrator

Step 14:

In the **OAuth Flows and External Client App Enhancements** section **Enable Client Credentials Flow** and add username.

OAuth Flows and External Client App Enhancements

Enable Client Credentials Flow

* Run As (Username)

ajp@force.com

Step 15:

In the **App Authorization** section, under **OAuth(Open Authorization) Policies**, click **Expire refresh token after a specific time**.

Step 16:

Configure token expiration settings:

- **Refresh Token Validity Period:** Enter 365
- **Refresh Token Validity Unit:** Select **Day(s)**

Step 17:

In the **Session Timeout in Minutes** box, enter **15**.

Step 18:

Click **Save**.

App Authorization

Refresh Token Policy

- Refresh token is valid until revoked
- Immediately expire refresh token
- Expire refresh token after specific time
- Expire refresh token if not used for specific time

* Refresh Token Validity Period

365

* Refresh Token Validity Unit

Day(s)

* IP Relaxation

Enforce IP restrictions

- Enable Single Logout
- High Assurance Session Required

Session Timeout In Minutes ¹

1 - 1440

Step 19:

Enable Allow Access to External Client App Consumer Secret via REST API

The screenshot shows the Salesforce 'External Client App Settings' page. A search bar at the top left contains the text 'external'. A red box highlights the search bar, with a red arrow pointing to the 'Settings' link in the left-hand navigation menu. Another red box highlights the 'Settings' link, with a red arrow pointing to the 'Allow access to External Client App consumer secrets via REST API' toggle switch, which is currently turned 'On'. A third red box highlights the toggle switch, with a red arrow pointing to the 'On' label. The page title is 'External Client App Settings' and the sub-header is 'External Client Apps'. The main content area contains two sections: 'Allow access to External Client App consumer secrets via Metadata API' and 'Allow access to External Client App consumer secrets via REST API'. The 'Allow access to External Client App consumer secrets via REST API' section has a toggle switch that is turned 'On'.

Your external client app is now ready to use.

How to Find Client ID (Consumer Key)

- type **external** in quick find search bar and click **external client app manager**
- under **External Client App Name** locate the app you created earlier and click it.

Setup Home Object Manager

external

External Client App Manager

External Client App Name	Contact Email	App Authorization
1 Testing_JWT	agudr...@gmail.com	Admin approved users are pre-authorized

Didn't find what you're looking for? Try using Global Search.

- under settings tab click **OAuth Settings** then you can the view your **client key** and **client secret** after the verification process.

Manage External Client Apps

Testing_JWT

Contact Email: agudr...@gmail.com

App Authorization: Admin approved users are pre-authorized

Type: Local

App Status: **Enabled**


Policies **Settings** Package Defaults

Configure policies to customize the external client app and plugins for this Salesforce organizati

App Policies

▼ OAuth Settings

App Settings

[Consumer Key and Secret](#) 

* Callback URL

http://localhost:1717/OauthRedirect

* OAuth Scopes



Verify Your Identity

You're trying to **access an external client app**. To make sure your Salesforce account is secure, we have to verify your identity.

Enter the verification code we emailed to ag*****@***il.com.

Verification Code

Back

Verify

[Resend Code](#)

Converting server.crt to JKS Keystore File

To use the certificate with Salesforce, convert it to a Java KeyStore (JKS) format.

Step 1:

Important step to successfully integrate into SIEM

Clone the server.key file and save it as server.pem.

In Linux

```
cp server.key server.pem
```

Note: Step 2 - 4 are **Optional - (use for salesforce to salesforce integration)**

Step 2:

Create a PKCS12 keystore file (minimum password length: 6 characters):

```
openssl pkcs12 -export -in server.crt -inkey server.pem -out keystore.p12
```

Step 3:

Convert the PKCS12 file to JKS format:

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

You will be prompted to create a password. Remember this password for future use.

Step 4:

Change the default alias (Salesforce doesn't support alias "1"):

```
keytool -keystore servercert.jks -changealias -alias 1 -destalias <name of certificate>
```

Required fields for JWT Authentication Integration:

- JWT Authentication Client Key Path (full file folder path of server.pem not in root directory)
 - ex: Users/jdoe/JWT/server.pem

- Username (can be found in users)
 - example format: ADMIN-3dvj@force.com
- Client ID (Consumer Key)
 - example format:
3MVxxxxtCx.CV6cbh7fSpKs_5iexxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxZKbaepcxIJUhO1
- Instance URL
 - example format: <https://company.my.salesforce.com>
- API Version
 - in quick find type **Apex Classes**

Action	Name ↑	Namespace Prefix	Api Version	Status
Edit Security	DeveloperEditionUtils	devedapp	64.0	Active
Edit	DeveloperEditionUtilsTest	devedapp	64.0	Active
Edit Security	PostInstallScript	devedapp	64.0	Active
Edit	PostInstallScriptTest	devedapp	64.0	Active

Provide this required fields to [CyTech Support](#).

Reference Link:

[Create an External Client App in Your Org | Salesforce DX Developer Guide |](#)

[Salesforce Developers](#)

If you need further assistance, kindly contact our support at support@cytechint.com for prompt assistance and guidance.

Revision #20

Created 11 February 2026 13:14:08 by Benjie Janlay Jr.

Updated 21 May 2026 14:05:58 by Benjie Janlay Jr.