

# Create an External Client App Using Metadata API

In this project, you'll learn how to create an external client app using Salesforce Metadata API. Instead of using Salesforce Setup, you'll configure and deploy the app through Salesforce CLI and Visual Studio Code. You'll set up your development environment, create an SFDX project, configure OAuth settings, and deploy the app to your Trailhead Playground. By the end, you'll have a functional external client app visible in the External Client App Manager.



## Requirements

### 1. Development Environment

- **Operating System:** Windows, macOS, or Linux
- **Software Requirements:**
  - **Salesforce CLI** (Installed and configured)
  - **Visual Studio Code (VS Code)** with **Salesforce Extension Pack**
  - Internet connection to access **Salesforce Trailhead Playground**

### 2. Salesforce Configuration

- **Enable Dev Hub:**
  - Navigate to **Setup** → **Dev Hub**
  - Click **Enable Dev Hub**
- **Authorize Dev Hub from CLI**

- Run:

```
Sh sf org login web --set-default-dev-hub --alias ecaViaMetadata --
instance-url https://<my_domain>
```

- **Verify My Domain**
  - Check **Setup** → **My Domain** and use the correct instance URL

### 3. SFDX Project Setup

- **Create SFDX Project**

- Run:

```
sh
```

```
sf project generate --name ecaViaMetadata --template standard
```

- **Modify sfdx-project.json**
  - Update sfdcLoginUrl to your Salesforce **My Domain**
- **Modify project-scratch-def.json**
  - Add the following features:

```
"features": ["EnableSetPasswordInApi", "ExternalClientApps",
"ExtlClntAppSecretExposeCtl"]
```

### 4. Metadata Files for External Client App

- **Create a Manifest (package.xml)**

```
xml
```

```
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
  <types>
    <members>*</members>
    <name>ExternalClientApplication</name>
  </types>
  <version>61.0</version>
</Package>
```

- **Create External Client App Metadata (ecaViaMetadata.eca-meta.xml)**
- **Set Up OAuth Plugin Configuration**

**Global OAuth Settings** (ecaViaMetadataGlbOAuth.ecaGlbOAuth-meta.xml)

**OAuth Settings** (ecaViaMetadataSettings.ecaOAuth-meta.xml)

## 5. Deployment & Verification

- **Deploy the App**

```
sh
sf project deploy start --manifest package.xml --target-org <username>
```

- **Retrieve Metadata After Deployment**

```
sh
sf project retrieve start --manifest package.xml --target-org <username>
```

- **Confirm in Salesforce:**

- Open **External Client App Manager** and verify **ecaViaMetadata** is listed
- Ensure the policies file is generated

## 6. Testing & Integration

- **Use OpenID Connect Playground** to test OAuth flow
- **Collect OAuth credentials:**
  - **My Domain**
  - **Consumer Key & Secret**
  - **Callback URL**

## Prepare Your Environment

### Learning Objectives

In this project, you'll:

- Install development tools.
- Create an external client app using Metadata API.

The first External Client Apps project used Salesforce Setup to create and interact with your external client apps. In this project, you're going to configure an app on your Trailhead Playground with Metadata API. Using Salesforce CLI, you create the files that make up an external client app in an SFDX Project and deploy them to your Trailhead Playground.

Most of the work you do to create an external client app via Metadata API takes place in Visual Studio Code (VS Code), but some changes need to be made to your Trailhead Playground. There are also a few steps you need to take to set up your local environment before you can get started.

## Launch Your Trailhead Playground

You complete this hands-on project on your own computer and your personal Salesforce environment. Get your Trailhead Playground now by first logging in to Trailhead, and then clicking the **Launch** button at the bottom of this page. After you complete the project steps in your playground, click **Verify step** at the bottom of the page.

## Set the Stage

Before we can connect your computer and your Trailhead Playground, you need to take a few steps to prepare them both. You'll turn the playground into a Dev Hub and download some applications to configure an SFDX project on your computer.

## Enable Dev Hub on Your Trailhead Playground

- From Setup in your Trailhead Playground, enter **Dev Hub** in the Quick Find box and select **Dev Hub**.
- Click **Enable Dev Hub**.

## Install Salesforce CLI and VS Code

Visual Studio Code is the go-to code editor for Salesforce developers. It's free, open-source, and available for Windows, Linux, and macOS. This editor has easy-to-install extensions for syntax highlighting, code completion, and more.

- Install [Salesforce CLI](#) on your computer so you can interact with your Trailhead Playground.
- Download and install the latest version of [Visual Studio Code](#) for your operating system. If you already have Visual Studio Code installed, there's no need to reinstall it.
- Launch Visual Studio Code.
- On the left toolbar, click the **Extensions** icon.
- Search for Salesforce Extension Pack and click **Install**.
- Now that your environment is setup, let's create the external client app.

## Create an External Client App

### Create an SFDX Project

Now that your environment is set up, it's time to create your external client app. To do that, you need an SFDX project.

- Create a folder called `ECA Metadata` on your computer where you want to create the app.
- Open the folder in VS Code.
- Open the Terminal in VS Code at the folder you created.
- Run this command in the Terminal to create a Salesforce DX project called `ecaViaMetadata` with the standard template.

```
sf project generate --name ecaViaMetadata --template standard
```

### Authorize Your Dev Hub Org

Connect the SFDX project on your computer to the Trailhead Playground Dev Hub using the connected app. You'll need your domain for this step. To find your domain in Setup, enter `My Domain` in the Quick Find box then click **My Domain**.

- Open the SFDX project ECA Metadata folder in VS Code.

- In the Terminal in VS Code, run this command after replacing <my domain> with your org's domain.

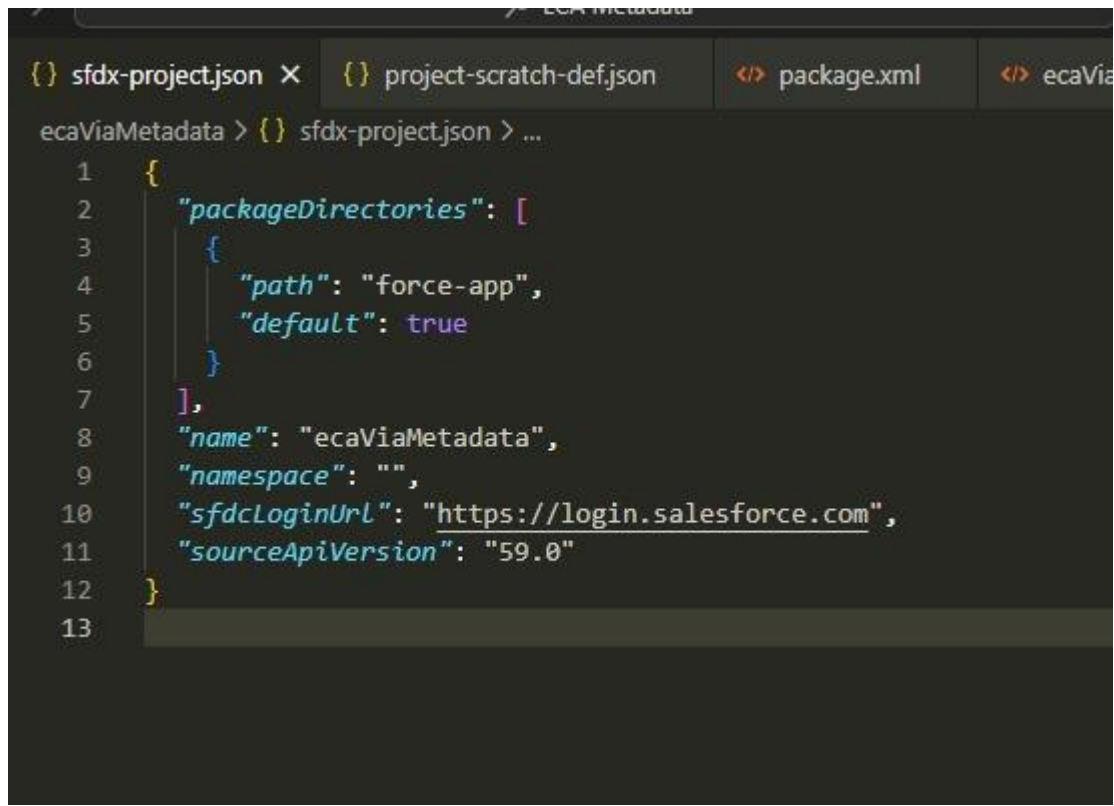
```
sf org login web --set-default-dev-hub --alias ecaViaMetadata --instance-url https://<my domain>
```

- Log in with your Trailhead Playground credentials in the web page that opens.
- Allow access to the org.

## Configure the SFDX Project for an External Client App

- 1) Open the `sfdx-project.json` file in VS Code.
- 2) Update the `sfdcLoginUrl` parameter to your domain.

To find your domain in Setup, enter **My Domain** in the Quick Find box then click **My Domain**.



```
ecaViaMetadata > {} sfdx-project.json > ...
1  {
2    "packageDirectories": [
3      {
4        "path": "force-app",
5        "default": true
6      }
7    ],
8    "name": "ecaViaMetadata",
9    "namespace": "",
10   "sfdcLoginUrl": "https://login.salesforce.com",
11   "sourceApiVersion": "59.0"
12 }
13
```

- 3) Expand the directory called *config* and open the scratch org definition file, which is called *project-scratch-def.json*.

Add **ExternalClientApps** and **ExtlClntAppSecretExposeCtl** to the features setting.

**"features": ["EnableSetPasswordInApi", "ExternalClientApps", "ExtlClntAppSecretExposeCtl"],**



```
ecaViaMetadata > config > {} project-scratch-def.json > ...
1  {
2    "orgName": "Demo company",
3    "edition": "Developer",
4    "features": ["EnableSetPasswordInApi"],
5    "settings": {
6      "lightningExperienceSettings": {
7        "enableS1DesktopEnabled": true
8      },
9      "mobileSettings": {
10       "enableS1EncryptedStoragePref2": false
11     }
12   }
13 }
14
```

- 4) Create a **package.xml** manifest file in the project directory.

- 5) Add this content to your **package.xml** file.

<?xml version="1.0" encoding="UTF-8"?>

<Package xmlns="http://soap.sforce.com/2006/04/metadata">

<types>

<members>\*</members>

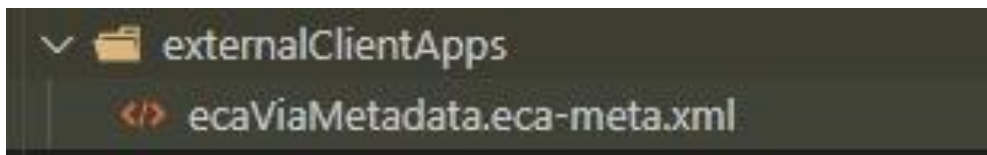
<name>ExternalClientApplication</name>

</types>

<version>61.0</version>

</Package>

- 6) In the force-app/main/default directory, create a folder called **externalClientApps**.
- 7) Add a file to the externalClientApps folder called **ecaViaMetadata.eca-meta.xml**. This will be the header file for your external client app.



- 8) Add this content to your header file.

```
<?xml version="1.0" encoding="UTF-8"?>
<ExternalClientApplication xmlns="http://soap.sforce.com/2006/04/metadata">
  <contactEmail>eca_metadata@example.com</contactEmail>
  <description>External client app Metadata API creation</description>
  <distributionState>Local</distributionState>
  <isProtected>false</isProtected>
  <label>ecaViaMetadata</label>
  <orgScopedExternalApp>00DdL00000Kf7wc:ecaViaMetadata</orgScopedExternalApp>
</ExternalClientApplication>
```

At this point your external client app is technically complete. You could deploy this configuration, and it would show up in the External Client App Manager on your playground as a basic external client app. However, to make a useful app, you need a plugin. Let's configure an OAuth plugin similar to the one we made in the first module. An OAuth plugin that is configured for the OAuth 2.0 Web Server flow requires a couple of edits and a couple new files.

## Enable and Configure the OAuth Plugin

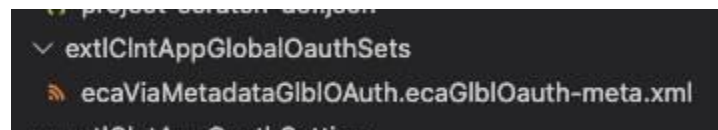
Unlike connected apps, which combine all configurations in a single file, external client apps include two settings files and a policies file. The Global OAuth



Settings file includes those sensitive fields like OAuth consumer key and consumer secret that should be protected. The OAuth Settings file includes all of the less-sensitive configurations for an external client app. There is no need to create a policies file while configuring an external client app for the web server flow, because policies are configured based on the settings files. The OAuth Policies file is generated when you deploy the external client app.

## Create a Global OAuth Settings File

- 1) In the force-app/main/default directory, create a folder called extlCIntAppGlobalOAuthSets.
- 2) Add a file to the extlCIntAppGlobalOAuthSets folder called ecaViaMetadataGlbIOAuth.ecaGlbIOAuth-meta.xml.



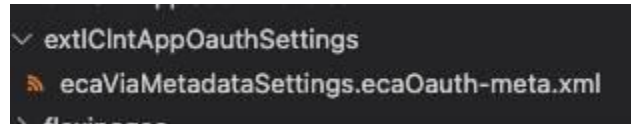
- 3) Open ecaViaMetadataGlbIOAuth.ecaGlbIOAuth-meta.xml in VS Code and add this content to it.

```
ecaViaMetadata > force-app > main > default > extlCIntAppGlobalOAuthSets > ecaViaMetadataGlbIOAuth.ecaGlbIOAuth-meta.xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <ExtlCIntAppGlobalOAuthSettings xmlns="http://soap.sforce.com/2006/04/metadata">
3      <callbackUrl>https://openidconnect.herokuapp.com/callback</callbackUrl>
4      <consumerKey>3MVG9GCMQoQ6rpzRRncum79jlnFv2GiSN0YrHzcSyyf4yqSmqf_sxKuhsQLtb5KcBtXkaQ1wk3.Drt.2RYtH
5      <externalClientApplication>ecaViaMetadata</externalClientApplication>
6      <isClientCredentialsFlowEnabled>false</isClientCredentialsFlowEnabled>
7      <isCodeCredFlowEnabled>false</isCodeCredFlowEnabled>
8      <isCodeCredPostOnly>false</isCodeCredPostOnly>
9      <isConsumerSecretOptional>false</isConsumerSecretOptional>
10     <isDeviceFlowEnabled>false</isDeviceFlowEnabled>
11     <isIntrospectAllTokens>false</isIntrospectAllTokens>
12     <isNamedUserJwtEnabled>false</isNamedUserJwtEnabled>
13     <isPkceRequired>false</isPkceRequired>
14     <isRefreshTokenRotationEnabled>false</isRefreshTokenRotationEnabled>
15     <isSecretRequiredForRefreshToken>true</isSecretRequiredForRefreshToken>
16     <isSecretRequiredForTokenExchange>false</isSecretRequiredForTokenExchange>
17     <isTokenExchangeEnabled>false</isTokenExchangeEnabled>
18     <label>ecaViaMetadataglobalset</label>
19     <shouldRotateConsumerKey>false</shouldRotateConsumerKey>
20     <shouldRotateConsumerSecret>false</shouldRotateConsumerSecret>
21 </ExtlCIntAppGlobalOAuthSettings>
22
```

- 4) Save the Global OAuth Settings file.

## Create an OAuth Settings File

- In the force-app/main/default directory, create a folder called extlCIntAppOAuthSettings.
- Add a file to the extlCIntAppOAuthSettings folder called ecaViaMetadataSettings.ecaOAuth-meta.xml.



- Open ecaViaMetadataSettings.ecaOAuth-meta.xml in VS Code and add this content to it.

```
ecaViaMetadata > force-app > main > default > extlCIntAppOAuthSettings > </> ecaViaMetadataSettings.ecaOAuth-meta.xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <ExtlCIntAppOAuthSettings xmlns="http://soap.sforce.com/2006/04/metadata">
3    <commaSeparatedOAuthScopes>Api, Web, OpenID</commaSeparatedOAuthScopes>
4    <externalClientApplication>ecaViaMetadata</externalClientApplication>
5    <label>ECA via Metadata OAuth Settings</label>
6    <oauthLink>00DdL00000Kf7wc:888dL000000EaOb</oauthLink>
7  </ExtlCIntAppOAuthSettings>
8
```

- Save the OAuth Settings file.

## Reference Settings Files in the Header File

Now that you have created the two settings files, you need to incorporate them into the external client app. The Header file is a list of all the files that make up the external client app.

- 1) Open the package.xml manifest file.
- 2) Add an entry for each of the three OAuth files.

```
> package.xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Package xmlns="http://soap.sforce.com/2006/04/metadata">
3      <types>
4          <members>*</members>
5          <name>ExternalClientApplication</name>
6      </types>
7      <types>
8          <members>*</members>
9          <name>ExtlClntAppOAuthSettings</name>
10     </types>
11     <types>
12         <members>*</members>
13         <name>ExtlClntAppGlobalOAuthSettings</name>
14     </types>
15     <types>
16         <members>*</members>
17         <name>ExtlClntAppOAuthConfigurablePolicies</name>
18     </types>
19     <version>61.0</version>
20 </Package>
```

- 3) Save the package.xml manifest file.

## Deploy Your External Client App

Now that your external client app is properly configured, deploy the app. The policies file is created on deployment, so after you deploy, retrieve the external client app to pull the generated file to your SFDX project.

- 1) Run this command to deploy the external client app. Replace <username> with the username you used to log in when you authorized the Dev Hub.

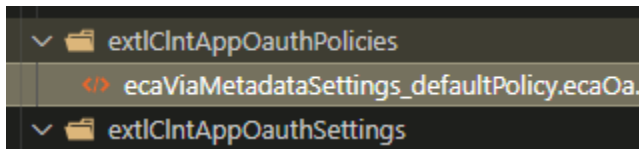
```
sf project deploy start --manifest package.xml --target-org <Username>
```

2) Retrieve the external client app from your Trailhead Playground. Replace <username> with the username you used to log in when you authorized the Dev Hub.

```
sf project retrieve start --manifest package.xml --target-org <username>
```

## Verify Your App

After successfully deploying and retrieving the external client app, you should see a policies file in your SFDX project where there wasn't one before.



Also, you can open the External Client App Manager and see a new external client app called ecaViaMetadata. If you're feeling adventurous, you could collect your My Domain and the OAuth consumer key and secret, and plug it all into the OpenID Connect Playground to walk through the OAuth Web Server flow..