

Securing MuleSoft API using Salesforce OAuth2 (OpenID Connect)

Author: [Uresh Kuruhuri](#)
Senior Consultant
Siddhi Worx LLC

Implementation	3
Step 1: Create (parent) Connected App with OpenID Connect scope in Salesforce.	3
Step 2: Create a Salesforce Client provider under Anypoint Access Management.	6
Step 3: Design and implement an API	9
Step 4: Apply the OpenID Connect token enforcement policy on the api gateway	10
Step 5: Request access to the API	13
Step 6: Generate access token	16
Step 7: Use access token to access the API using the API proxy	19
Additional Resources	20

Implementation

Step 1: Create (parent) Connected App with OpenID Connect scope in Salesforce.

Even though we mention Salesforce as identity provider in this context, it is not enabled explicitly in this implementation. We need to create a Connected App in the Salesforce side. Let's call it MuleSoft (for the sake of understanding, also can be referred to as parent connected app). In practice, the MuleSoft parent connected app sends a request to the dynamic client registration endpoint to create a child connected app. The child connected apps (or the client app created in MuleSoft, mentioned in Step #5 above) are needed for consumers to access data.

The first step is to [create a connected app in Salesforce](#). Also check this additional link on [configuring the basic information for the app](#).

1. Login to Salesforce and go to Setup. In Quick Find search box, type and select App Manager.
2. Click New Connected App.
3. In the screen, enter the required details as below

SETUP
App Manager

New Connected App

To publish an app, you need to be using a Developer Edition organization with a namespace prefix chosen.

Basic Information

Connected App Name

API Name

Contact Email

Contact Phone

Logo Image URL
[Upload logo image](#) or [Choose one of our sample logos](#)

Icon URL

Info URL

Description

API (Enable OAuth Settings)

Enable OAuth Settings ☐

- a. **Connected App Name:** Enter a name for the connected app (consider this one as parent app). e.g. MuleSoft.

- b. **API Name:** Usually this auto populates based on the connected app name. You are allowed to change it to the required value.
 - c. **Contact Email:** Email contact for this connected app.
 - d. **Contact Phone:** Phone number contact for this connected app (optional)
 - e. **Logo Image URL:** You can upload an image (optional)
 - f. **Enable OAuth Settings:** Enable the check mark field.
4. Proceed to select and add the necessary OAuth scopes and other settings as required. Check the following screenshot:

The screenshot shows the 'SETUP App Manager' interface. Under the 'API (Enable OAuth Settings)' section, the following settings are visible:

- Enable OAuth Settings:** Checked (blue checkmark).
- Enable for Device Flow:** Not checked.
- Callback URL:** A text field containing the letter 'a'.
- Use digital signatures:** Not checked.
- Selected OAuth Scopes:** A list of available scopes with 'Add' and 'Remove' buttons. The selected scopes are:
 - Access your basic information (id, profile, email, address, phone)
 - Allow access to your unique identifier (openid)
 - Perform requests on your behalf at any time (refresh_token, offline_access)
- Available OAuth Scopes:** A list of available scopes including:
 - Access and manage your Eclair data (eclair_api)
 - Access and manage your Salesforce CDP Ingestion API data (cdp_ingest_api)
 - Access and manage your Wave data (wave_api)
 - Access and manage your data (api)
 - Access custom permissions (custom_permissions)
 - Allow access to Lightning applications (lightning)
 - Allow access to content resources (content)
 - Full access (full)
 - Provide access to custom applications (visualforce)
 - Provide access to your data via the Web (web)
- Require Secret for Web Server Flow:** Checked.
- Require Secret for Refresh Token Flow:** Checked (labeled 'c').
- Introspect All Tokens:** Checked.
- Configure ID Tokens:** Checked.
- Token Valid for:** 30 minutes (labeled 'd').
- ID Token Audiences:** A text field containing 'CONSUMER_KEY'.
- Include Standard Claims:** Checked.
- Include Custom Attributes:** Not checked.
- Include Custom Permissions:** Not checked.
- Enable Asset Tokens:** Not checked.

- a. **Enter a callback URL:** <https://login.salesforce.com/services/oauth2/callback>
- b. **Selected OAuth Scopes:** Select the scopes per the screenshot. If the screenshot is not visible, add the following scopes:
 - Access your basic information (id, profile, email, address, phone)*
 - Allow access to your unique identifier (openid)*
 - Perform requests on your behalf at any time (refresh_token, offline_access)*
- c. **Select relevant check boxes:**
 - Require Secret for Web Server Flow*
 - Require Secret for Refresh Token Flow*
 - Introspect All Tokens*
 - Configure ID Token and Include Standard Claims*

Once you save the settings the connected app is created.

However, make sure the relevant policy is associated with the connected app. In order to do so, view (**App Manager** → MuleSoft connected app → click dropdown arrow at the end and click View) the connected app.

Now click **Manage** and then click **Edit Policies** button (see screenshot below). Select the option that you want to use and proceed with the remaining steps.

Connected App Edit

MuleSoft

Version 1
Description

Basic Information

Start URL Mobile Start URL

OAuth Policies

Permitted Users ☒ All users may self-authorize
Admin approved users are pre-authorized

Enable Single Logout ☐

IP Relaxation Enforce IP restrictions

Refresh Token Policy: ☒ Refresh token is valid until revoked
☐ Immediately expire refresh token
☐ Expire refresh token if not used for Day(s)
☐ Expire refresh token after Day(s)

Session Policies

Timeout Value --None--

☐ High assurance session required

In this example , **All users may self-authorize** option was used. Also, I left the default **Enforce IP restrictions** as is. You may change this per your requirements.

Click **Save** down below.

Once the setup is complete, the connected app is created and ready. The connected app after creation would look similar to the screenshot below.

SETUP Manage Connected Apps

Connected App Name
MuleSoft

[Back to List: Custom Apps](#)

[Edit](#) [Delete](#) [Manage](#)

Version 1.0
API Name MuleSoft
Created Date 7/1/2021 9:28 AM
By: Uresh Kuruhuri
Contact Email
Contact Phone
Last Modified Date 7/2/2021 9:04 AM
By: Uresh Kuruhuri
Description
Info URL

API (Enable OAuth Settings)

Consumer Key Zuq

Consumer Secret [Click to reveal](#)

Selected OAuth Scopes ☒ Access your basic information (id, profile, email, address, phone)
☒ Perform requests on your behalf at any time (refresh_token, offline_access)
☒ Allow access to your unique identifier (openid)

Enable for Device Flow ☐

Require Secret for Refresh Token Flow ☒

Token Valid for 0 Hour(s)

Include Custom Permissions ☐

Require Secret for Web Server Flow ☒

Introspect All Tokens ☒

Include Custom Attributes ☐

Enable Single Logout ☐ Single Logout disabled

Initial Access Token for Dynamic Client Registration

Initial Access Token [Regenerate](#)

Initial Access Token Value [Click to reveal](#)

Configure ID Token

Token Valid for 30 Minutes

Include Standard Claims ☒

Include Custom Permissions ☐

Custom Connected App Handler

Apex Plugin Class

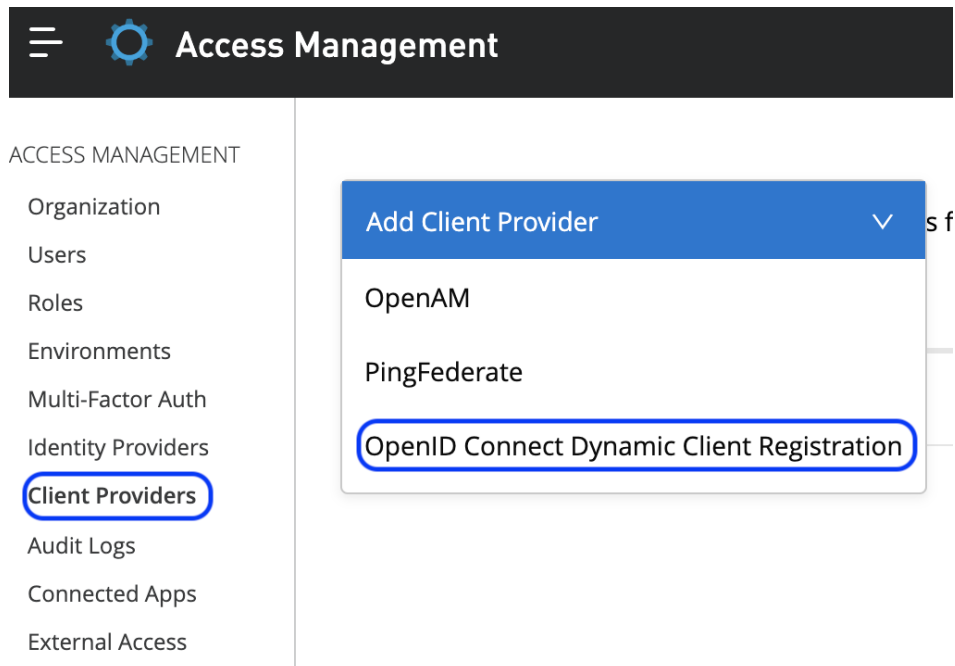
Run As

Verify the highlighted sections in this screenshot. Also take a note of the **Initial Access Token for Dynamic Client Registration**. This will be used in the following setup in Anypoint Platform. This completes the setup on the Salesforce side.

Step 2: Create a Salesforce Client provider under Anypoint Access Management.

Now it is time to initiate the **dynamic client registration** process in the Anypoint Platform. The user needs to have admin access to do this. To do this follow the steps below:

1. Login to Anypoint platform.
2. Navigate to Access Management → Client Providers and click Add Client Provider.
3. Select OpenID Connect Dynamic Client Registration.



4. It opens up the following page to enter the relevant details.

The screenshot shows the 'Access Management' interface in Salesforce. The left sidebar contains links for 'ACCESS MANAGEMENT' (Organization, Users, Roles, Environments, Multi-Factor Auth, Identity Providers, Client Providers, Audit Logs, Connected Apps, External Access), 'SETTINGS' (Runtime Manager, Flow Designer), and 'SUBSCRIPTION' (Runtime Manager, Object Store). The main content area is titled 'Dynamic Client Registration' and includes the following fields and options:

- Issuer** (a): A text field containing a domain name.
- Client Registration URL*** (b): A text field containing a URL ending in '/services/oauth2/register'.
- Authorization Header** (c): A text field with masked characters and a 'Show' button.
- Advanced settings** (d, e): A section with two checkboxes: 'Disable server certificate validation' (unchecked) and 'Enable client deletion in Anypoint Platform' (checked). Below it, 'Enable client deletion and updates in IdP' is also checked.
- Token Introspection Client** section:
 - Client ID*** (f): A text field containing a value ending in 'soV'.
 - Client Secret** (g): A text field with masked characters and a 'Show' button.
- OpenID Connect Authorization URLs** section:
 - Authorize URL*** (h): A text field containing a URL ending in '/services/oauth2/authorize'.
 - Token URL*** (i): A text field containing a URL ending in '/services/oauth2/token'.
 - Token Introspection URL*** (j): A text field containing a URL ending in '/services/oauth2/introspect'.
- Buttons**: 'Cancel' and 'Save' buttons at the bottom.

The below urls and related OpenID connect metadata can be obtained by using the url [\\$ISSUER/.well-known/openid-configuration](#) where the \$ISSUER is the issuer of the OpenID Connect related connected app.

In this case, if you have a domain based salesforce e.g. it would be something like this <https://uresh.my.salesforce.com/.well-known/openid-configuration>

Or in most cases it would be <https://login.salesforce.com/.well-known/openid-configuration>

- a. **Issuer:** Enter the issuer name e.g. <https://uresh.my.salesforce.com>
 - b. **Client Registration URL :** Enter the client registration url. e.g. <https://uresh.my.salesforce.com/services/oauth2/register>
 - c. **Authorization Header:** Here you have to enter the initial access token in this format [Bearer <initial-access-token>](#). e.g. Bearer 0a3772898Abcj8.....fhasdfo97231D
 - d. **Enable client deletion in Anypoint platform:** This is optional. This enables you to delete the client apps created for the API in Anypoint platform.
 - e. **Enable client deletion and updates in IdP:** This is optional. This enables you to delete/update the client app created in the Identity Provider. In this example [Salesforce](#).
 - f. **Client ID:** This is the consumer key that is generated in the Salesforce for the parent connected app.
 - g. **Client Secret:** This is the consumer secret that is generated in the Salesforce for the parent connected app.
 - h. **Authorize URL:** The OpenID authorization url as is available from the metadata. e.g. <https://uresh.my.salesforce.com/services/oauth2/authorize>
 - i. **Token URL:** The OpenID token generation url as is available from the metadata. e.g. <https://uresh.my.salesforce.com/services/oauth2/token>
 - j. **Token Introspection URL:** The OpenID token introspection URL as is available from the metadata. e.g. <https://uresh.my.salesforce.com/services/oauth2/introspect>
5. Click **Save**.

Note: In order to use this client provider, it needs to be associated with an environment. Also, once associated with an environment, this client provider is not automatically applied to the existing API's.

In order to apply this client provider to one of the environments, do the following:

1. Navigate to **Access Management** → **Environments**.
2. Select an environment. It opens the following dialog box.

Edit environment

Name
Sandbox

Client ID
[Redacted]

Client Secret
[Redacted] Show

Client provider (optional) ⓘ
If none selected, Anypoint will remain as the client provider for this environment.

Select ▼
Salesforce OIDC Remove

Delete Environment Cancel Update

3. Under the **Client provider (optional)** field, select the newly created identity provider.
4. Click **Update**.

Step 3: Design and implement an API

For keeping the brevity of this article, API design and implementation is not shown here. Anyway, the important thing here is that you apply the OAuth2.0 security scheme to your API. Here below is the sample security scheme for your reference.

```
#!/RAML 1.0 SecurityScheme
type: OAuth 2.0

description: |
  This API supports OAuth 2.0 for authenticating all API requests.
describedBy:
  queryParameters:
    access_token:
      description: |
        Used to send a valid OAuth 2 access token.
      type: string
  responses:
    400:
      description: Invalid token.
    401:
      description: |
        Unauthorized or Connection error when connecting to the authorization
server.
    403:
      description: |
        Forbidden, invalid client application credentials.
    500:
      description: |
        Bad response from authorization server, or WSDL SOAP Fault error.
  settings:
    authorizationUri: https://uresh.my.salesforce.com/services/oauth2/authorize
    accessTokenUri: https://uresh.my.salesforce.com/services/oauth2/token
    authorizationGrants: [ authorization_code ]
```

Step 4: Apply the OpenID Connect token enforcement policy on the api gateway

Once the Dynamic Client Registration is complete and the client provider is associated with an environment, it is now available to be applied to the API's.

The option of selecting the **OpenID Connect token enforcement policy** would only be available after a client provider is associated with the environment.

In order to apply for the existing API's you have to go the API Manager, select the api and update the Client provider field to the newly enabled client provider.

However, in this example we add the api to the API manager and the client provider is automatically selected.

Follow the steps below to accomplish this

1. Login to Anypoint Platform.
2. Go to API Manager.

3. Click Manage API → Manage API from Exchange.

Manage API from Exchange

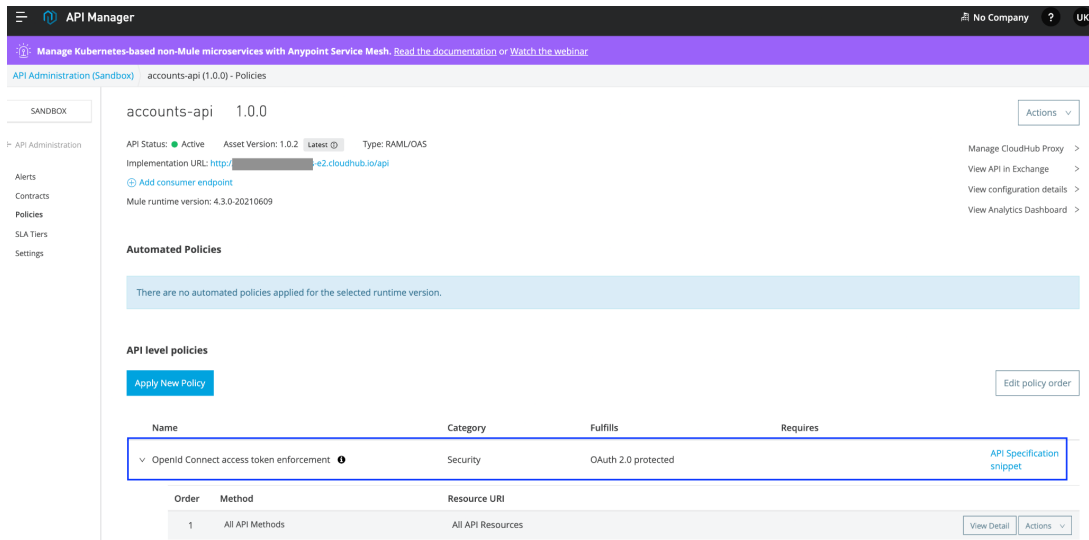
API Configurations

API name:	accounts-api		a
Asset type:	RAML/OAS		
API version:	1.0.0	View API in Exchange	b
Asset version:	1.0.2		c
Managing type:	☐ Basic Endpoint ☒ Endpoint with Proxy		d
	Proxy deployment target: ☒ CloudHub ☐ Hybrid		e
Mule version:	☒ Mule 4 Recommended ☐ Mule 3 or below		f
Implementation URI:	https://abc.com/api		g
Client provider: ⓘ	Salesforce OIDC		h
TLS Context for outbound traffic:	⊕ Add TLS Context Used to secure outbound traffic		
Path:	/		
Advanced options >			
Cancel		Save	

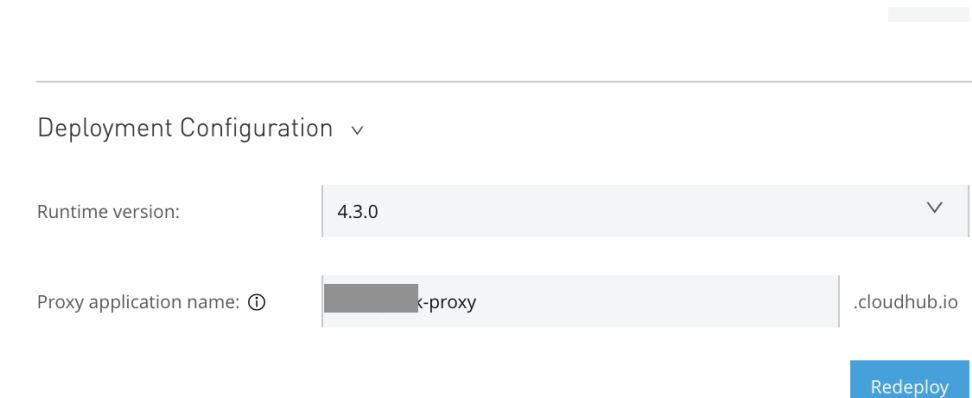
In the page now enter the following information per the example screenshot below **for your API**.

- API name:** Search and select the api you want to manage.
- API version:** Select the API version you want to manage.
- Asset version:** Select the asset version of the API you want to manage.
- Managing Type:** Select Endpoint with Proxy if you want to manage using an API gateway.
- Proxy deployment target:** By default it is Cloudhub. If you have to manage the API in on-prem instance, select Hybrid.
- Mule version:** Select *Mule 4* or as applicable.
- Implementation URI:** Provide the implementation url for the API. e.g. `https://abc.com/api`

- h. **Client provider:** Select the identity provider created in **Step 2**.
4. Click **Save**.
5. Now go to **Policies** and click *Apply New Policy*.
6. Select **OpenID Connect token enforcement policy**. Refer to the screenshot below.



7. Once the policy is applied and set, Select **Settings** again.
8. **Deployment configuration** section is now available to deploy the API gateway (proxy app).



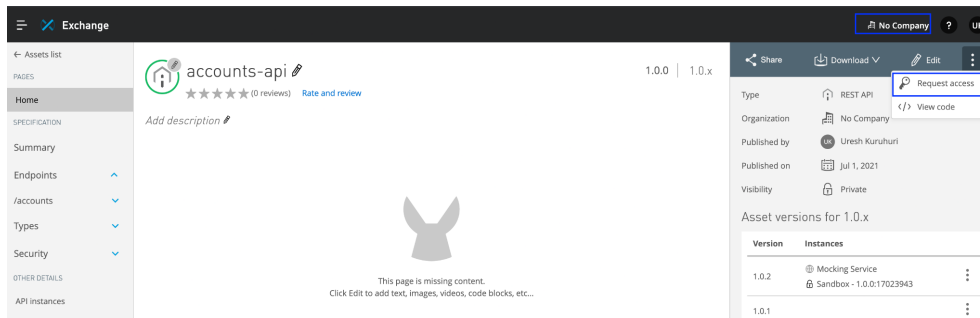
9. Select the **Runtime version** as application. In this example it is **v4.3.0**.
10. Give a name to the proxy application.
11. Click **Deploy/Redeploy** to have the API gateway (proxy) deployed in the Cloudhub.

This completes applying the policy

Step 5: Request access to the API

Now that the main configuration is set up, we need to request access to the API. As you may know it, we request it from the Anypoint Platform. By doing so, this process automatically creates a client connected app in Salesforce (following the settings set in the parent Mulesoft app) because of the DCR process.

1. Login to Anypoint Platform and go to Exchange.
2. Search the API and open the API page.



3. Click the vertical ellipsis icon to open the extra menu items and click Request access.
4. It shows up the following dialog

Request access

API Instance

1.0.0:17023943

Application

Select application

SLA tier
(Optional)

accounts-api-client

Create a new application

Cancel

Request access

5. In the API Instance, select the instance of the app that we want to request access to.

6. In the Application dropdown, click Create a new application. It opens up a new dialog to enter the details.

Create new application

Application Name

Enter application name **a**

Description
(Optional)

Write a description here... **b**

Application URL
(Optional)

https://yourcompany.com/applicationURL **c**

OAuth 2.0 Grant type

☐ Implicit Grant **d**

☐ Authorization Code Grant

☐ Refresh Token (Requires one of Authorization Code Grant)

OAuth 2.0 redirect URIs ⓘ
(Optional)

https://yourcompany.com/callback **e**

☐ Automatically register the redirect URIs for API Notebook and API Console

Cancel

Create

- a. **Application name:** Enter the application name. e.g. accounts-api-client.
- b. **Description:** This is optional. You can enter the description that you would like for this client app.
- c. **Application URL:** This is optional as well. You can leave it or enter the application url.
- d. **OAuth 2.0 Grant Type:** Select the applicable grant type for the OAuth access for this client app. I selected the **Implicit** and **Authorization Code Grant** (this needs **Refresh Token** grant as well.)
- e. **OAuth 2.0 redirect URIs:** This is optional, but in our case we would use the url as: <https://login.salesforce.com/services/oauth2/success>. Remember this has to be used as is while retrieving the access token.
7. Click **Create**. This step automatically creates a client connected app in Salesforce as well (let's call this API related connected app or child connected app). Here below are screenshots from Salesforce/MuleSoft for this app.

Manage Connected Apps

Connected App Name: **accounts-api-client**

Back to List: Custom Apps

Version: 1.0
API Name: accounts_api_client
Created Date: 7/1/2021 3:14 PM
Contact Email: By: Automated Process
Contact Phone: contact@autocreated.1625177681507
Last Modified Date: 7/1/2021 3:14 PM
By: Automated Process
Description: Info URL

API (Enable OAuth Settings)

Consumer Key: 179
Consumer Secret: Click to reveal

Selected OAuth Scopes: Access your basic information (id, profile, email, address, phone)
Access and manage your data (api)
Perform requests on your behalf at any time (refresh_token, offline_access)
Allow access to your unique identifier (openid)

Callback URL: https://login.salesforce.com/services/oauth2/success

Enable for Device Flow: ☐
Require Secret for Refresh Token Flow: ☐
Token Valid for: 0 Hour(s)
Include Custom Permissions: ☐
Require Secret for Web Server Flow: ☐
Introspect All Tokens: ☐
Include Custom Attributes: ☐
Enable Single Logout: Single Logout disabled

Custom Connected App Handler

Apex Plugin Class: Run As

If you observe, it automatically adds another OAuth scope **Access and manage your data (api)**. The below screenshot shows the client application created in MuleSoft.

Exchange

My applications list

accounts-api-client

This is a test client app.

Client ID: 3MVG9_J_oWktdLrIE_HMjgHqyr7sW2iwQ2vMqRMxN1ES.EQIWUSzS_VeKawUg9q...
Client Secret: Show

Application URL: -
Redirect URIs: https://login.salesforce.com/services/oauth2/success

Grant Types: Authorization code, Implicit, Refresh token

Sandbox

accounts-api 1.0.x
1.0.0:17023943

The *Client ID* and *Client Secret* shown in the MuleSoft app page are exactly same as *Consumer Key* and *Consumer Secret* respectively, for the connected app created in Salesforce

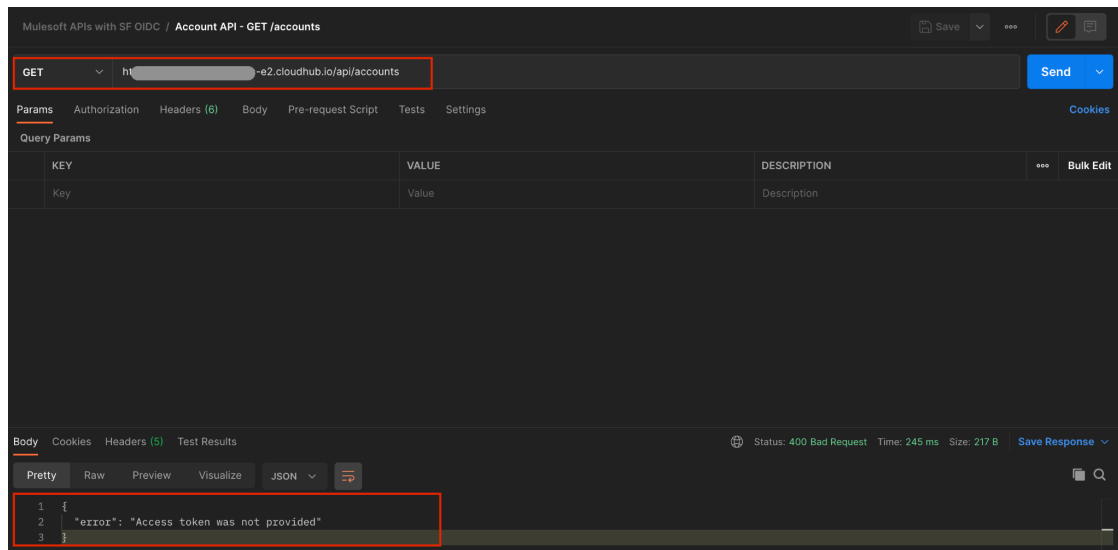
As of this step, the main setup is done.

Step 6: Generate access token

This part is done using the Postman app here **for demo purposes only**. *You may have to work on code client or as is the case may be.*

First let's check if the API implementation is accessible. Once the API gateway is created and deployed, the direct access to the API should not be available.

Try accessing one of the resources and you should see an error message as with HTTP 400 error as Bad request. Here is a sample screenshot



Let's get an access token.

We have to enter the following values in the Configure New Token utility under Postman per the screenshot. Open a tab and set the address pointing to a resource using the API gateway url (proxy).

MuleSoft APIs with SF OIDC / Account API Gateway - GET /accounts

GET http://proxy.us-e2.cloudhub.io/accounts Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Type OAuth 2.0

The authorization data will be automatically generated when you send the request. [Learn more about authorization](#)

Add authorization data to Request Headers

Configure New Token

Configuration Options Advanced Options

Token Name **a** SF-IDP-AccessToken

Grant Type **b** Authorization Code

Callback URL **c** https://login.salesforce.com/services/oauth2/success

Auth URL **d** https://uresh.my.salesforce.com/...

Access Token URL **e** https://uresh.my.salesforce.com/...

Client ID **f** [Redacted]

Client Secret **g** [Redacted]

Scope **h** e.g. read:org

State [Redacted]

Client Authentication **h** Send client credentials in body

Clear cookies

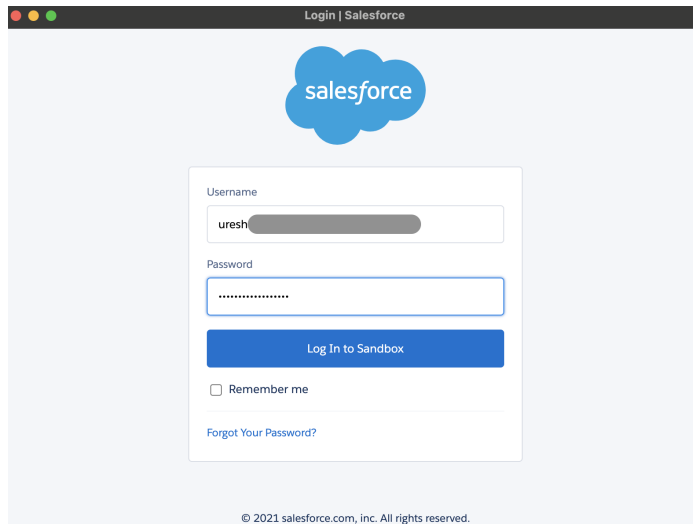
Get New Access Token

Fill in the values for the relevant fields as suggested below:

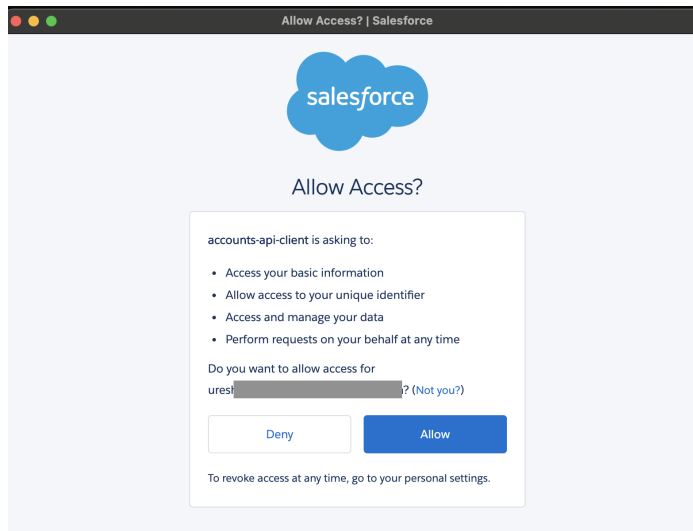
- Token Name:** You can set any name as you like and permitted.
- Grant Type:** You can select one of the grant types that were set to the connected app created as a part of requesting access to the api in the previous step. In this example, we will go with [Authorization Code](#).
- Callback URL:** This url has to be exactly same as the one set in the connected app create for the api. This has to be the redirect url that was in the OAuth 2.0 redirect URIs in the previous step. In this example it is <https://login.salesforce.com/services/oauth2/success>
- Auth URL:** It should be the same as the one retrieved in the [OpenID Connect metadata](#). In this example, it was set to <https://uresh.my.salesforce.com/services/oauth2/authorize>.
- Access Token URL:** It should be the same as the one retrieved in the OpenID Connect metadata. In this example, it was set to <https://uresh.my.salesforce.com/services/oauth2/token>
- Client ID:** The Client ID that was generated for the connected app for the api (i.e. accounts-api-client) in the previous step: Request access to the API.
- Client Secret:** The Client secret that was generated for the connected app for the api (i.e. accounts-api-client) in the previous step: Request access to the API.
- Client Authentication:** Select Send client credentials in body option.

Click **Get New Access Token**.

It prompts for Salesforce Login. Enter your Salesforce credentials.



Once the authentication is successful, it prompts to authorize access as below



Click **Allow**.

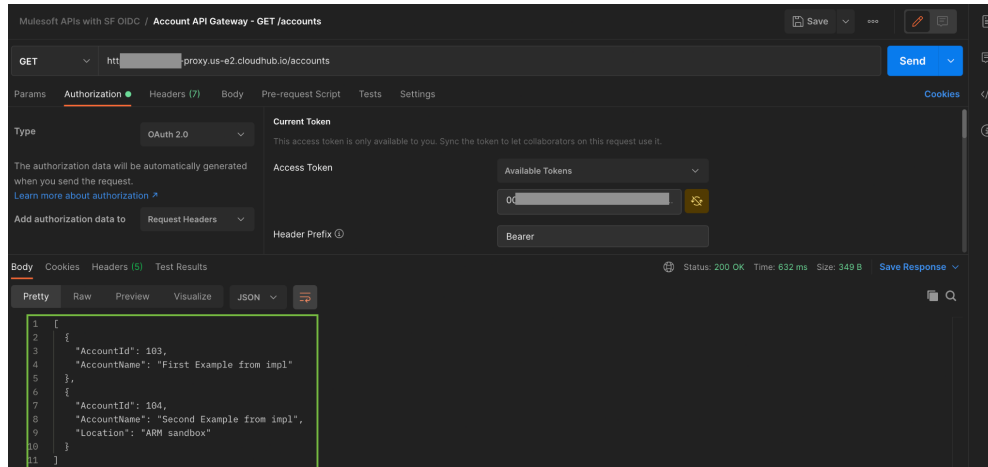
Once it is done, Postman utility gets the new token and shows the **Manage Access Tokens** dialog as below. Click **Use Token**.

Step 7: Use access token to access the API using the API proxy

Now that we have the access token we can use it to access the resource(s) of the API using the API gateway (proxy app).

Continuing in this example, in Postman, the access token is already set (when Use Token was clicked).

Click Send to see the API JSON payload returned in the body section as shown in the screenshot below.



This concludes the implementation.

Additional Resources

- [Connecting Your APIs with MuleSoft and Salesforce Identity](#)
- [Integrate Service Providers as Connected Apps with OpenID Connect](#)
- [Salesforce: Single Sign-On](#)
- [Salesforce: Configure Basic Connected App Settings](#)
- [Salesforce: Manage OAuth Access Policies for a Connected App](#)
- [Salesforce: OpenID Connect Token Introspection](#)
- [MuleSoft: Configure OpenID Connect Client Provider](#)