API Security : Securing APIs with OAuth (3-legged)

Duration : 30 mins
Persona : API Team/Security

Use case

You have a set of APIs that are consumed by trusted partners. You want to secure those APIs using a three legged OAuth (Authorization Code flow).

How can Apigee Edge help?

Apigee Edge quickly lets you secure your APIs using out of the box OAuth policies. OAuth defines token endpoints, authorization endpoints, and refresh endpoints. Apps call these endpoints to get access tokens, to refresh access tokens, and, in some cases, to get authorization codes. These endpoints refer to specific OAuth 2.0 policies that execute when the endpoint is called.

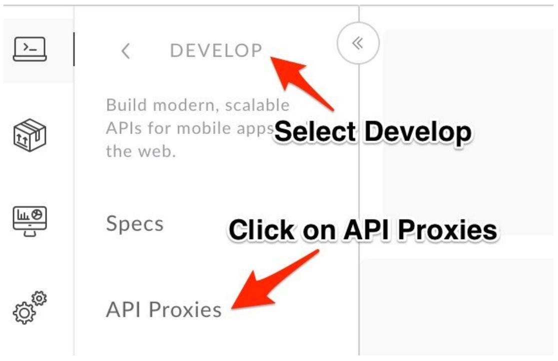
Authorization code is one of the most commonly used OAuth 2.0 grant types. The authorization code flow is a "three-legged OAuth" configuration. In this configuration, the user authenticates himself with the resource server and gives the app consent to access his protected resources without divulging username/passwords to the client app. This grant type is intended for apps that are written by third-party developers who do not have a trusted business relationship with the API provider. For example, developers who register for public API programs should not generally be trusted. With this grant type, the user's credentials on the resource server are never shared with the app.

Pre-requisites

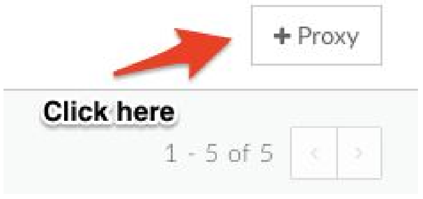
- You have an OAuth API proxy in Apigee Edge. If not, jump back to “API Development - Create a Reverse Proxy” lab.
- You have the following created on Apigee Edge - an API Product, a Developer and an App. If not, jump back to “API Publishing - Packaging APIs” lab.

Instructions

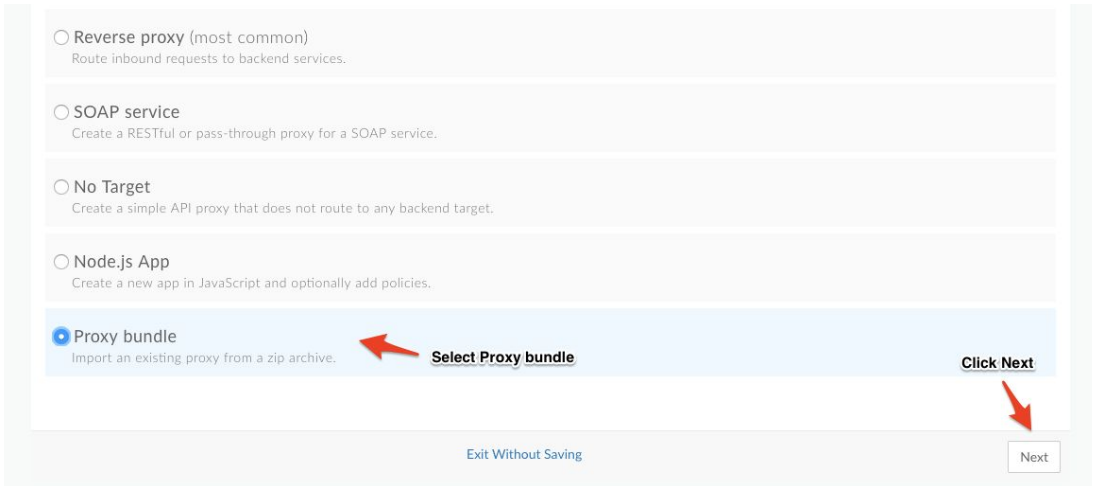
- Go to https://apigee.com/edge and log in. This is the Edge management UI.
- Select **Develop > API Proxies** in the side navigation menu



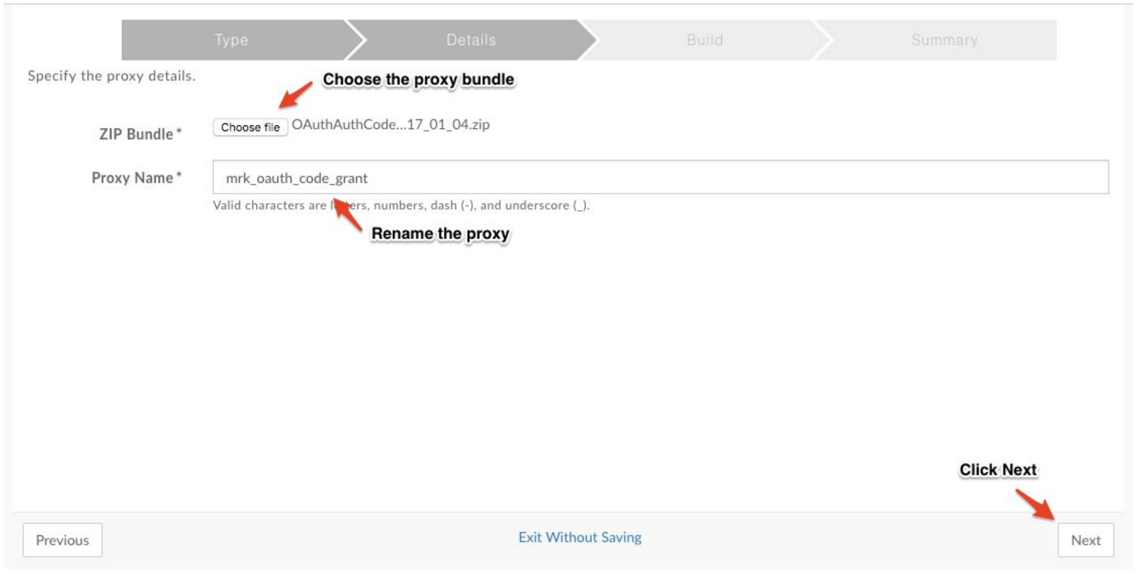
- Download the api proxy bundle from here
- Click the **+ Proxy** button



- Select **Proxy Bundle** option and click Next.


Reverse proxy (most common)
Route inbound requests to backend services.
SOAP service
Create a RESTful or pass-through proxy for a SOAP service.
No Target
Create a simple API proxy that does not route to any backend target.
Node.js App
Create a new app in JavaScript and optionally add policies.
Proxy bundle
Import an existing proxy from a zip archive.
Select Proxy bundle
Click Next
Exit Without Saving
Next

- Now, choose the bundle that you downloaded earlier, rename the proxy in this format ***{your_initials}****_oauthcodegrant* and click Next and then Build



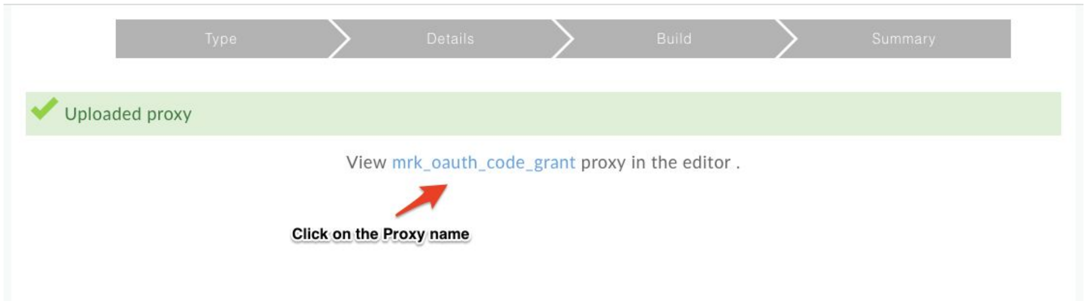
Specify the proxy details.

ZIP Bundle * OAuthAuthCode...17_01_04.zip

Proxy Name * mrk_oauth_code_grant

Valid characters are letters, numbers, dash (-), and underscore (_).

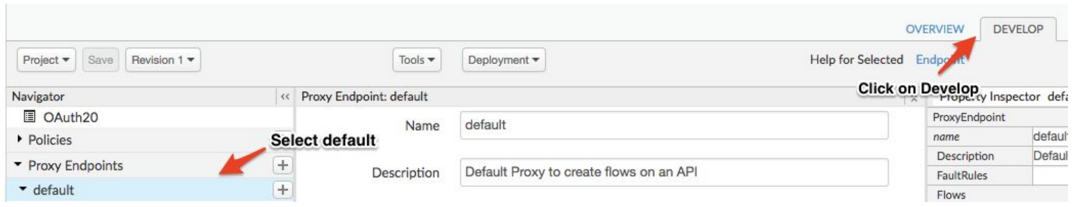
- Click on the API Proxy name.



Uploaded proxy

View mrk_oauth_code_grant proxy in the editor .

- Click on develop and select default

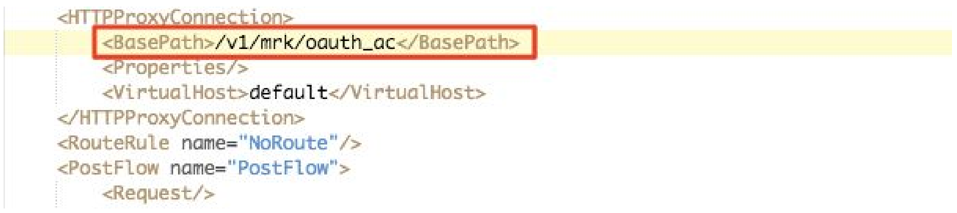


Proxy Endpoint: default

Name: default

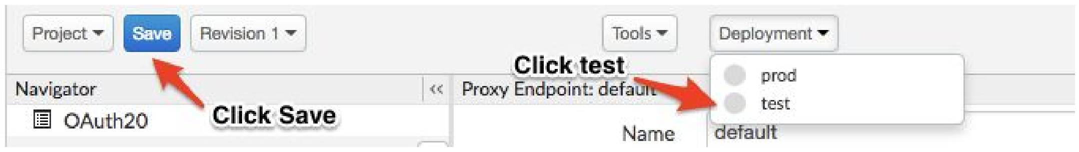
Description: Default Proxy to create flows on an API

- From the code editor, find the base path entry and update it so that it follows this format `/v1/{your_initials}/oauth_ac`

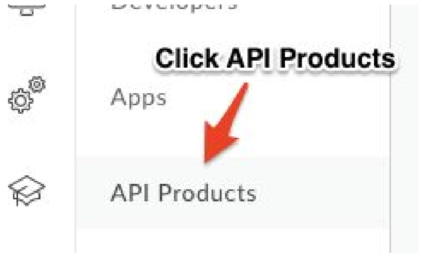


```
<HTTPProxyConnection>
    <BasePath>/v1/mrk/oauth_ac</BasePath>
    <Properties/>
    <VirtualHost>default</VirtualHost>
</HTTPProxyConnection>
<RouteRule name="NoRoute"/>
<PostFlow name="PostFlow">
    <Request/>
```

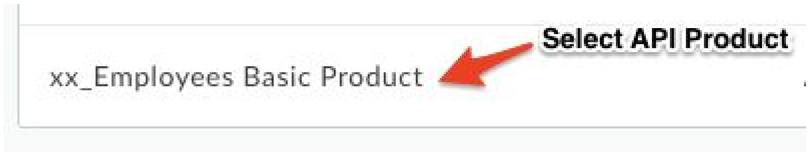
- Now Save the proxy and deploy it to test



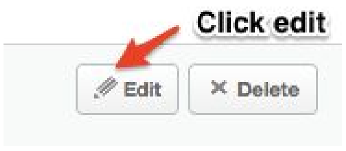
- Once the proxy is deployed, select **Publish > API Products** from the side navigation menu



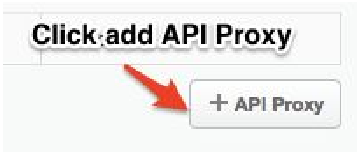
- Select the API Product that you created as a part of "Publishing APIs" lab.



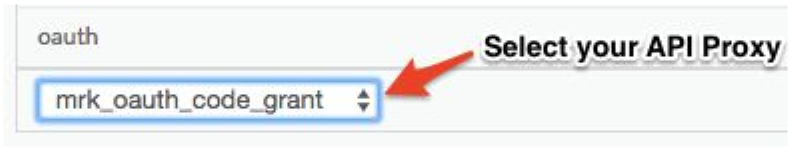
- Click on Edit



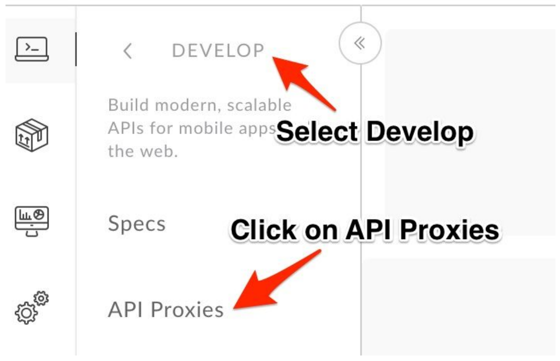
- Now under Resources > API Proxy section, click on Add API Proxy



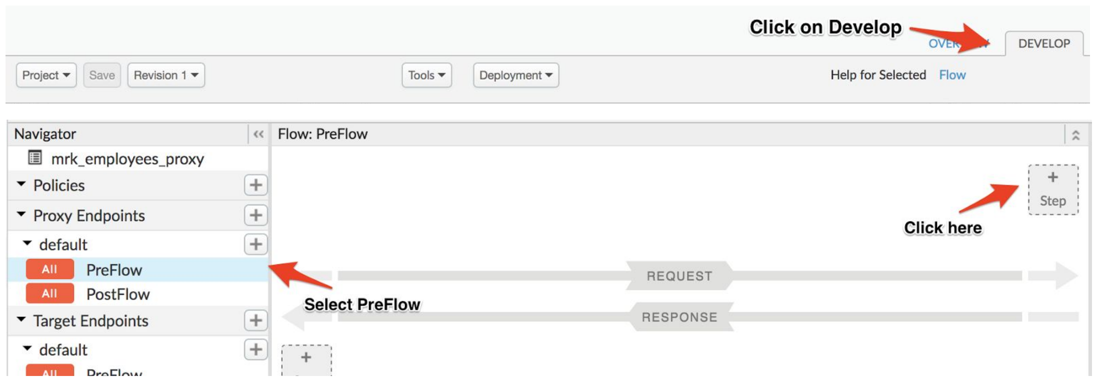
- Select the API Proxy and click save



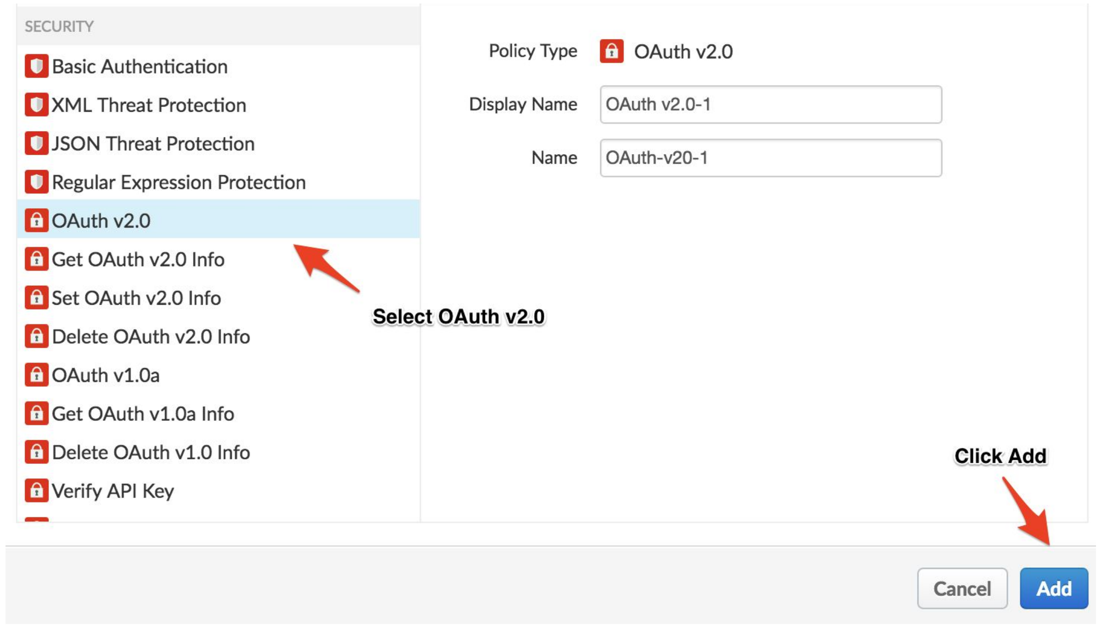
- Now. select **Develop > API Proxies** in the side navigation menu



- Click on the API proxy that you created in “Creating an API proxy” lab.
- Click on the **Develop** tab. Select **PreFlow** from the sidebar under **Proxy Endpoints** section



- Click on Add Step and in the dialog, select OAuth v2.0 from the Security section then click the Add button



Policy Type: OAuth v2.0

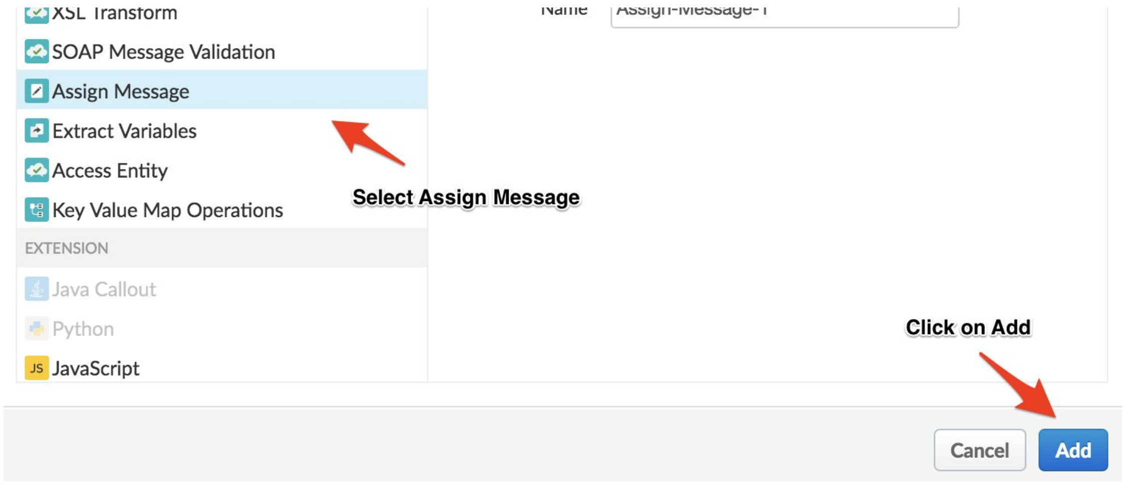
Display Name: OAuth v2.0-1

Name: OAuth-v20-1

- Click on the policy and in the code editor, paste the code give below

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
     <OAuthV2 async="false" continueOnError="false"
     enabled="true" name="OAuth-v20-1">
         <DisplayName>OAuth v2.0-1</DisplayName>
         <ExternalAuthorization>false</ExternalAuthorization>
         <Operation>VerifyAccessToken</Operation>
         <SupportedGrantTypes/>
         <GenerateResponse enabled="true"/>
</OAuthV2>
```

- Once again click on Add Step and in the dialog, select Assign Message policy from the Mediation section then click the Add button

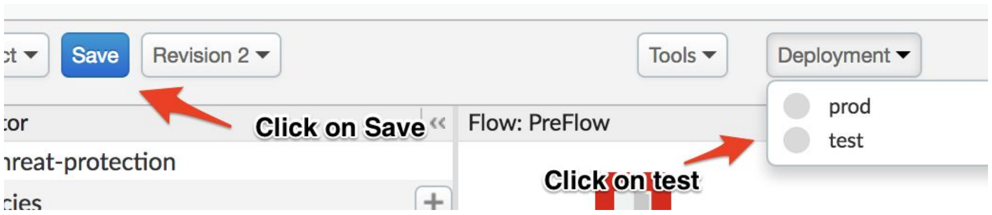


- Click on the policy and in the code editor, paste the code give below

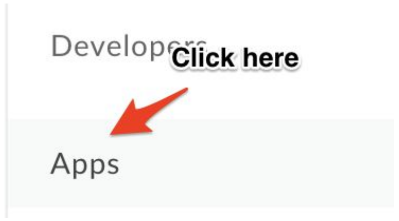
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AssignMessage async="false" continueOnError="false"
enabled="true" name="Assign-Message-1">
     <DisplayName>Assign Message-1</DisplayName>
     <Remove>
        <Headers>
           <Header name="Authorization"/>
        </Headers>
     </Remove>

<IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
    <AssignTo createNew="false" transport="http"
type="request"/>
</AssignMessage>
```

Note: You'll have to remove the Authorization header using the Assign Message policy because, the header might create some conflict in the target backend.

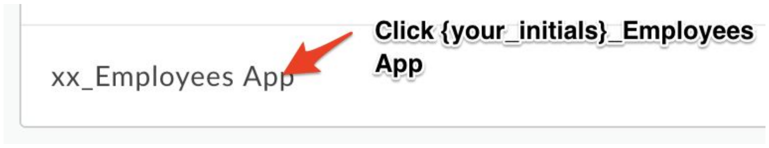
- **Save** the proxy and deploy it on the **test** environment



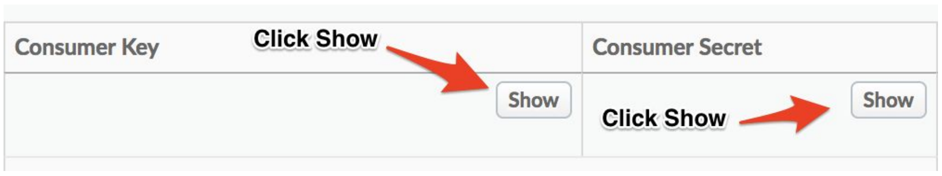
- *Congratulations!*...You've now successfully secured your APIs with OAuth 2.0
- Now, let's test it. To do that, we'd have to obtain the consumer key and secret for a particular app that is associated with a API Product containing the APIs that we created.
- Click **Publish > Apps** from the side navigation menu



- Select the app that you created in the Publishing APIs lab

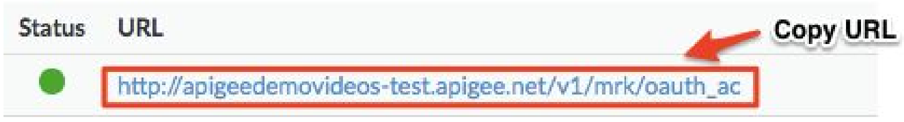


- Click on the show button under Consumer Key, Consumer Secret.
- Copy the values and store them somewhere safe



- Mac and Linux users, open Terminal and type the following command
  `echo -n <consumer_key>:<consumer_secret> | base64`
  Windows users, refer this link, or use this link to generate the value.

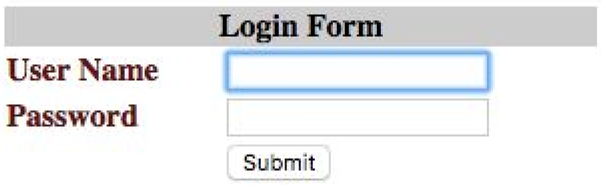
- Copy the URL for oauth API proxy.



http://apigeedemovideos-test.apigee.net/v1/mrk/oauth_ac

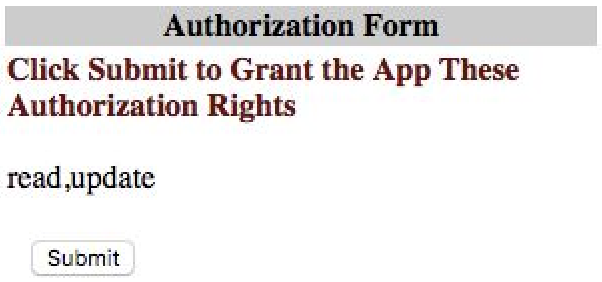
- First, you'll obtain an auth code which will be exchanged to obtain the access token. To obtain an auth code, you'll have to call the **`/authorize`** endpoint with your app's client id, a code response type and required scopes as query params
  Query param: `response_type=code, client_id=<your app's client id>, scope=READ,UPDATE`
  The final URL will look something like this -
  http://apigeedemovideos-test.apigee.com/v1/mrk/oauth_ac/authorize?client_id=<client_id>&response_type=code&scope=READ,UPDATE

- Make a call to this URL from your browser, it will then redirect you to a login page which will look like this.



  Use these details to login
  User Name: test
  Password: password

- After you're logged in, you should be able to see the consent page asking you to authorize the permissions

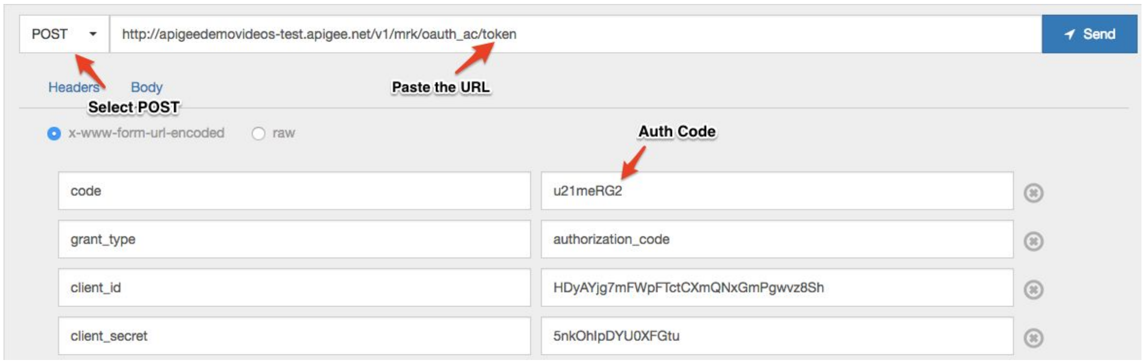


  Click on the submit button

- You will be redirected to the callback URL that you provided while creating the app with a query parameter i.e, code.
- To obtain an access token, we need to call the `/token` endpoint with the following body information passed in the **x-www-form-url-encoded** format

```
code : <auth code>
grant_type : authorization_code
client_id : <app’s client_id>
client_secret : <app’s client_secret>
```

- Now, let’s use the REST Client (https://apigee-rest-client.appspot.com/) to obtain an access token. Open the REST Client on a new browser window.
- Paste the URL with the /token endpoint, select POST and then fill up the body

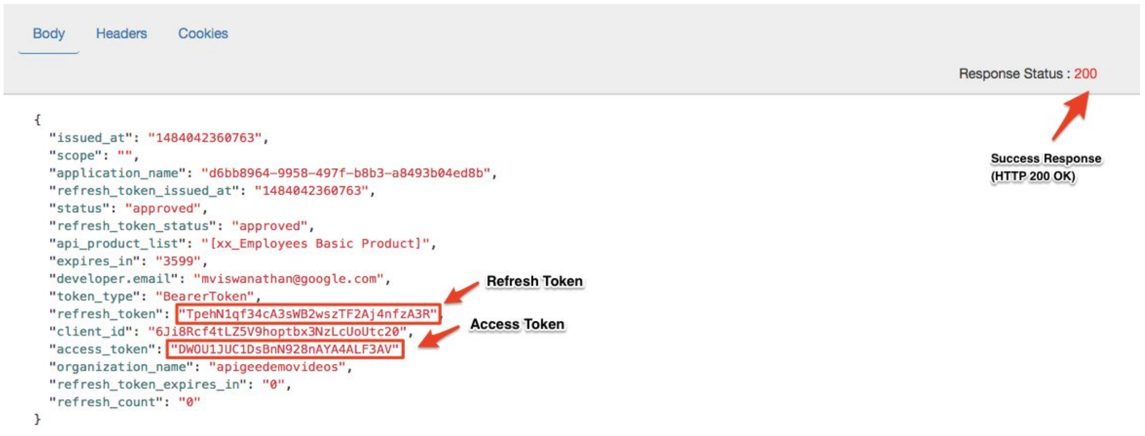


http://apigeedemovideos-test.apigee.net/v1/mrk/oauth_ac/token

x-www-form-url-encoded

code	u21meRG2
grant_type	authorization_code
client_id	HDyAYjg7mFWpFTctCXmQNxGmPgwvz8Sh
client_secret	5nkOhIpDYU0XFGtu

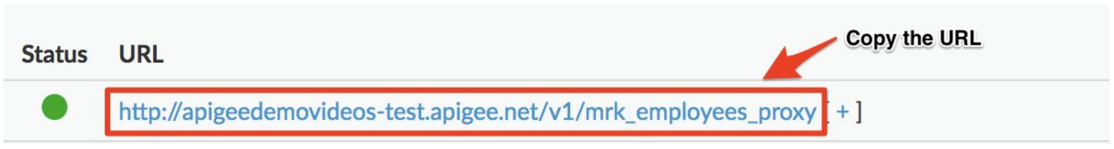
- Hit **Send** and you should see a response like this below. Then, copy the value for access token, refresh token and save them.



Response Status : 200

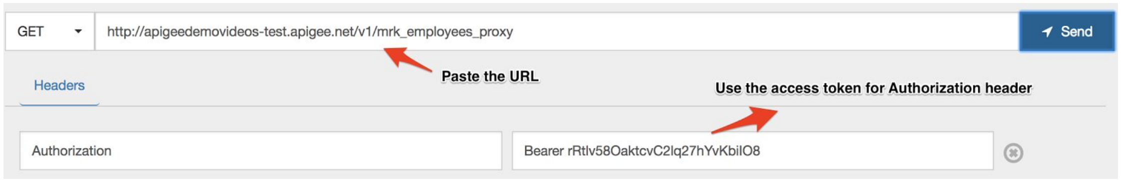
```
{
  "issued_at": "1484042360763",
  "scope": "",
  "application_name": "d6bb8964-9958-497f-b8b3-a8493b04ed8b",
  "refresh_token_issued_at": "1484042360763",
  "status": "approved",
  "refresh_token_status": "approved",
  "api_product_list": "[xx_Employees Basic Product]",
  "expires_in": "3599",
  "developer.email": "mviswanathan@google.com",
  "token_type": "BearerToken",
  "refresh_token": "TpehN1qf34cA3sWB2wszTF2Aj4nfzA3R",
  "client_id": "6Ji8Rcf4tLZ5V9hoptbx3NzLcUoUtc20",
  "access_token": "DWOU1JUC1DsBnN928nAYA4ALF3AV",
  "organization_name": "apigeedemovideos",
  "refresh_token_expires_in": "0",
  "refresh_count": "0"
}
```

- Now, you should be able to get the employees list using the access token that we just obtained. Copy the URL for the proxy you created earlier in this lab.



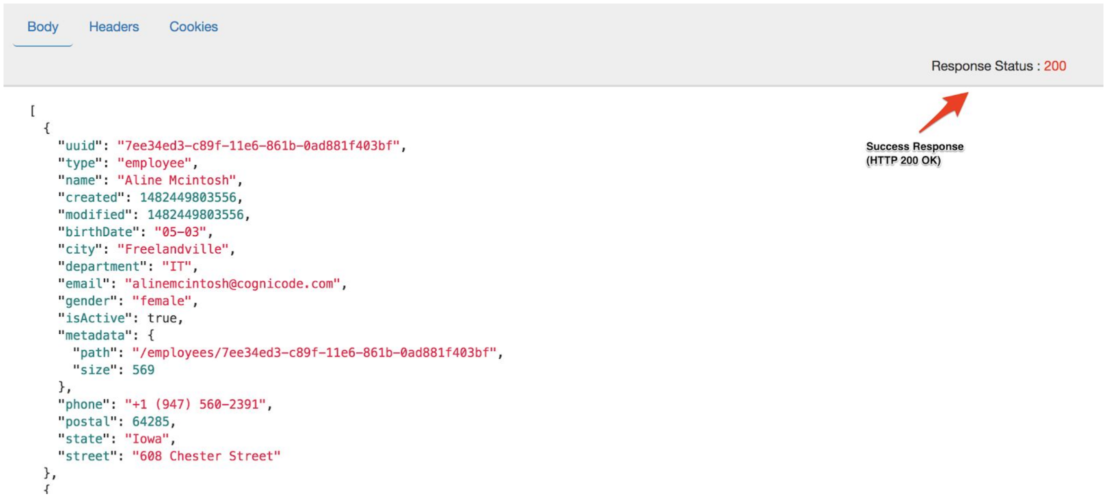
Status	URL
●	http://apigeedemovideos-test.apigee.net/v1/mrk_employees_proxy [+]

- Paste the URL in the Rest client and add the Authorization header. The value for Authorization header will be the access token that we obtained previously.
  `Authorization: Bearer <access_token>`



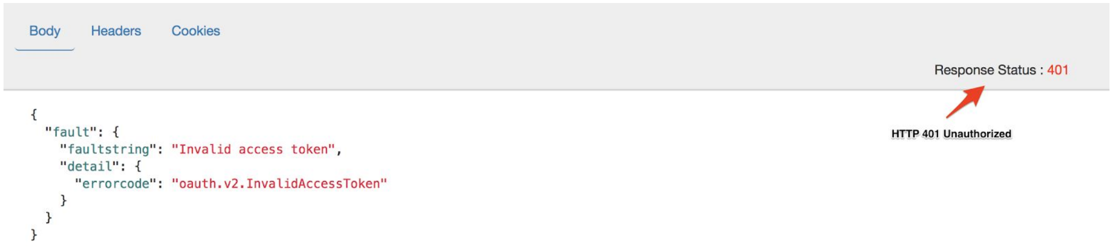
```
http://apigeedemovideos-test.apigee.net/v1/mrk_employees_proxy
Authorization    Bearer rRtlv58OaktcvC2lq27hYvKbilO8
```

- Hit **Send** and you should see a response like this below.



```
[
  {
    "uuid": "7ee34ed3-c89f-11e6-861b-0ad881f403bf",
    "type": "employee",
    "name": "Aline Mcintosh",
    "created": 1482449803556,
    "modified": 1482449803556,
    "birthDate": "05-03",
    "city": "Freelandville",
    "department": "IT",
    "email": "alinemcintosh@cognicode.com",
    "gender": "female",
    "isActive": true,
    "metadata": {
      "path": "/employees/7ee34ed3-c89f-11e6-861b-0ad881f403bf",
      "size": 569
    },
    "phone": "+1 (947) 560-2391",
    "postal": 64285,
    "state": "Iowa",
    "street": "608 Chester Street"
  },
  {
```

- And, if you remove the Authorization header and hit send, you will see a 401 Unauthorized status.



```
{
  "fault": {
    "faultstring": "Invalid access token",
    "detail": {
      "errorcode": "oauth.v2.InvalidAccessToken"
    }
  }
}
```

Lab Video

You can watch this short video to see how to implement 3 legged OAuth on Apigee Edge.
https://youtu.be/vPryGej4ydg

Earn Extra-points

Now that you've learned how to secure your API with OAuth 2.0, try to implement a refresh token flow which will give you an access token based in exchange for the refresh token.

Quiz

1. How do you revoke an access token?
2. Explain how apigee edge lets you integrate with an external IdP?

Summary

In this lab you learned how to secure your API using a three legged OAuth, obtaining an auth code, exchanging it for an access code and using that against your API.

References

- Link to Apigee docs page
    - OAuth 2.0: Configuring a new API proxy
      http://docs.apigee.com/api-services/content/understanding-default-oauth-20-configuration
    - Secure an API with OAuth - Authorization Grant Type
      http://docs.apigee.com/api-services/content/oauth-v2-policy-authorization-code-grant-type
- Link to Community posts and articles with topic as "OAuth 2.0"
- Search and Revoke tokens -
  https://community.apigee.com/articles/1571/how-to-enable-oauth-20-token-search-and-revocation.html

Rate this lab

How did you link this lab? Rate here.