Smart+Connected Digital Platform API Overview - Authentication and API Access using Python

Overview

Cisco Smart+Connected Digital Platform is a platform providing rich API for retrieving information from connected cities and device collections including sensor data, users, locations and capabilites. The API provides access to real time and historical data from the connected sensors and devices.

All Smart+Connected Digital Platform APIs (except authentication) require access tokens. Before any of these APIs can be used, you must authenticate with the Smart+Connected Digital Platform API to get the tokens needed to make successful calls.

Smart+Connected Digital Platform API Requirements

All Smart+Connected Digital Platform APIs, other than authentication, are protected and require two access tokens in order to use them and retrieve data. These tokens are:

1. api_access_token
2. app_access_token

Application developers integrating with the **Smart+Connected Digital Platform** must register their app via the Cisco **Smart+Connected Digital Platform Developers** portal. During registration, the system generates a **Client ID** and **Client Secret** pair, which are later used by the application to access the **Smart+Connected Digital Platform** authentication service. During this exercise, you use the provided **Client ID** and **Secret**, along with the provided Smart+Connected Digital Platform username and password, to authenticate and make an additional Smart+Connected Digital Platform API Request.

In this document, we will examine a small sample Python script that demonstrates how to perform **Smart+Connected Digital Platform** API authentication and make additional API calls. We'll walk through some of the key concepts and code details and use the access tokens returned by the authentiction call to make additional Smart+Connected Digital Platform API requests.

If you have Python 3 installed on your computer, you can try the code there as we go along.

Main Steps

We will make several API requests to get different types of data from the Smart+Connected Digital Platform APIs. Before we can make any API calls to retrieve data, we need to authenticate with the Smart+Connected Digital Platform to get the access tokens. Next, we will make another call to get account information for the current user. This will return two ids (user id and customer id) we will then use to make two more requests.

1. **Authenticate**
    - Authenticate using the **/login** Smart+Connected Digital Platform API by making a **POST Request** and submitting the required data in the post body
    - **Parse the response** to get the two tokens needed for all other API requests
2. **Retrieve User Information**
    - Make a **GET Request** to the **/accounts/username** Smart+Connected Digital Platform API
    - **Parse the response** to get the userid and customerid for the current user. These are required by some of the other Smart+Connected Digital Platform APIs and will be used in the last two steps.
3. **Retrieve Location Information**
    - Make a *GET Request** to the **locations/userinfo/user/<userId>** CDSmart+Connected Digital PlatformP API
4. **Retrieve Smart+Connected Digital Platform Capabilities Information**
    - Make a *GET Request** to the **capabilities/customer/** Smart+Connected Digital Platform API

The full python script can be found **here** if you want to try it out.

Steps

Step 1: Using the Smart+Connected Digital Platform Authentication API to request access tokens

> **Smart+Connected Digital Platform** Authentiction URL
> `http://10.10.20.6/apigw/devnetlabtokenapi/login`

There are five pieces of information that must be passed to the **Authentication API**:

1. Username (**username**) - This is the email address of the user logging in to the Smart+Connected Digital Platform API

2. Password (**password**) - This is the password associated with the username
3. Client Secret (**client_secret**) - This is the shared secret used by your application
4. Client ID (**client_id**) - This is the unique id for your application
5. Authorization Type (**grant_type**) - This will be set to 'client_credentials' for Smart+Connected Digital Platform API authentication For DevNet, these will be provided to you. For Smart+Connected Digital Platform Deployments, these can be obtained from the Smart+Connected Digital Platform Portal

Successful authentication will return two tokens:

1. API Token (**api_access_token**) - This is added as a request header with the following format
    - 'Authorization' : 'Bearer <api_access_token>'
2. APP Token (**app_access_token**) - This is added as a request header with the following format
    - 'WSO2-Authorization' : 'oAuth Bearer <app_access_token>'

Both of these are required to be passed as request headers to **all subsequent Smart+Connected Digital Platform API requests**.

Request

We need to build a request, specify the post data containing the four data items and make the Rest API POST request to Smart+Connected Digital Platform API.

The login URL is a special API that does not require access tokens. This API will return the access tokens needed for all other requests.

The format of this API URL is https://<**cdp-url**>/<**token-api-path**>/**login**

For this exercise, we will use this API URL:

```
http://10.10.20.6/apigw/devnetlabtokenapi/login
```

We will use the Python **urllib** module for the request and **json** module to perform these tasks.

```
import urllib.parse, urllib.request, json
```

We will then set the post data and specify the login url. ```

this dictionary contains the post body we will send

to the /login API

@note: these are examples and you need to replace them with your client id and secret

postData = { 'client*id':'51f38215eBAD4c118dcdbec86d77e574', 'client*secret':'352de2ed6f8c4478BAD6667F0809A2E4', 'grant*type':'client*credentials' }

this is the url we are using to authenticate with Smart+Connected Digital Platform

loginUrl = 'http://10.10.20.6/apigw/devnetlabtokenapi/login' ```

Next we will get the username and password and add it to the post data dictionary.

```
# get username and password
username = input('Enter username (email address): ')
password = input('Enter password: ')

# Add username/password to post data
postData['username'] = username;
postData['password'] = password;
```

Finally, we will make the request to the API. For urllib, we need to convert the dictionary to binary data to include with the post.

Note: You don't specify POST or GET when making a request with urllib. If you include data, the request will be a POST. If there is no data, the request will be a GET.

```
# urlencode the data so that symbols don't cause problems
data = urllib.parse.urlencode(postData)

# use UTF-8 encoding for POST data and responses
encoding = 'UTF-8'

# POST uses binary data, so encode it with the above encoding
binary_data = data.encode(encoding)

# urlopen/Request with data causes a post request
request = urllib.request.Request(loginUrl, binary_data)
response = urllib.request.urlopen(request)
```

Response

The response is a JSON object containing the tokens needed for subsequent API calls. We will convert the response to a dictionary so that we can retrieve and store the tokens.

```
# process the results and put into a JSON object/dictionary
results = response.read().decode(encoding)
responseDict = json.loads(results)
```

Retrieving Tokens from the Response

A successful login will result in the tokens being returned and available in our dictionary.

```
appAccessToken = responseDict['app_access_token']
apiAccessToken  = responseDict['api_access_token']
```

Since we will need to send these tokens as request headers in all future API calls, we can create a dictionary for our headers and store them there.

```
requestHeaders = {
    'WSO2-Authorization' : 'oAuth Bearer ' + appAccessToken,
    'Authorization' : 'Bearer ' + apiAccessToken
    'Accept': 'application/json'
}
```

Note: The added the 'Accept' header in the dictionary above. This was added to specify that we want JSON responses from the Smart+Connected Digital Platform API. We will use all three of these headers in subsequent Smart+Connected Digital Platform API calls.

Step 2: Retrieve User Information

In this step, we will use the access tokens we retrieved in **Step 1** to get additional information about the current user including their **UserId** and **CustomerID**.

The format of this API URL is https://<**cdp-url**>/<**api-path**>/**accounts/username**.

For this exercise, we will use this API URL:

```
http://10.10.20.6/apigw/devnetlabapi/cdp/v1
```

This API also requires a query parameter '?loginName=<username>' to specify the account information we want to retrieve.

Add the code below to your existing script from **Step 1** to make the next request.

Request

```
# specify the query param
queryParams =  urllib.parse.urlencode({'loginName': postData['username']})

# create the request URL
requestUrl = 'http://10.10.20.6/apigw/devnetlabapi/cdp/v1' + '/accounts/username?%s'
% queryParams;

# create the request
request = urllib.request.Request(requestUrl)

# add headers (for API authorization)
# these were retrieved from the response in Step 1
for k, v in requestHeaders.items():
  request.add_header(k, v)

# perform the request
response = urllib.request.urlopen(request)
```

Response

The response is a JSON object containing information about the current user. We will get two items from the reponse, UserID and CustomerID. We will convert the response to a dictionary so that we can retrieve these pieces of data.

```
# process the results and put into a JSON object/dictionary
results = response.read().decode(encoding)
responseDict = json.loads(results)
```

Retrieving UserID and CustomerID from the Response

The response dictionary now contains the UserID and CustomerID that we need.

```
userId = str(responseDict['id'])
```

The CustomerID is nested in another object, so we need to get that object and then get the CustomerID. In the Smart+Connected Digital Platform Account User Information response, the CustomerID is the 'id' of the 'parent' object in the dictionary.

```
parentInfo = responseDict['parentInfo']
if 'id' in parentInfo:
    customerId = str(parentInfo['id'])
```

That's it! We now have the access tokens we need for future Smart+Connected Digital Platform API calls and the UserID and CustomerID that we will use to get more details from Smart+Connected Digital Platform in the next two steps.

Step 3: Retrieving Location Information from the Smart+Connected Digital Platform API

In this step, we will retrieve the **Location Information** for the logged in user. This will return all of the Smart+Connected Digital Platform Locations that this user has access to.

> **Smart+Connected Digital Platform** Location URL
> `http://10.10.20.6/apigw/devnetlabapi/cdp/v1/locations/userinfo/user/<userId>`

Request

We need to build a new GET request that specifies they userId of the user whose location information we are querying.

The format of this API URL is https://<**cdp-url**>/<**token-api-path**>/**locations/userinfo/user/**

For this step, we will use this API URL:

```
http://10.10.20.6/apigw/devnetlabapi/cdp/v1/locations/userinfo/user/<userId>
```

Build the request and perform the GET

```
# make sure we have a userId from Step 2
if userId:
    # build the request URL, using the actual customerId in place of <customerId>
    requestUrl = baseUrl + '/capabilities/customer/' + customerId
    print('\nGetting **CAPABILITIES** Information (' + requestUrl + ')\n');
    request = urllib.request.Request(requestUrl)
    # create the request (not adding the 2nd data param means this is a GET request)
    request = urllib.request.Request(requestUrl)

    # add headers (for API authorization)
    for k, v in requestHeaders.items():
        request.add_header(k, v)

    # perform the request
    response = urllib.request.urlopen(request)
    results = response.read().decode(encoding)

    # create a dictionary from the results
    responseDict = json.loads(results)

    # print the results
    print(results)
else:
    print("error retrieving user information. 'userId' was not present")
```

That's it. We now have a list of locations available to this user that we can use as needed in other Smart+Connected Digital Platform APIs (to specify which devices to get real time data from for instance).

Next we will get information about what domains are available for from the Smart+Connected Digital Platform for the current customer and instance

Step 4: Retrieving Capabilities Information from the Smart+Connected Digital Platform API

In this step, we will retrieve the **Capabilities** of this Smart+Connected Digital Platform instance. This will provide us with information about the domains that are avaialble on a specific Smart+Connected Digital Platform instance for a specific customer.

> **Smart+Connected Digital Platform** Capabilities URL
> `http://10.10.20.6/apigw/devnetlabapi/cdp/v1/capabilities/customer/<customerId>`

Request

We need to build a new GET request that specifies they customerId we are querying.

The format of this API URL is https://<**cdp-url**>/<**token-api-path**>/**capabilities/customer/**

For this step, we will use this API URL:

```
http://10.10.20.6/apigw/devnetlabapi/cdp/v1/capabilities/customer/<customerId>
```

If you have Python 3 installed on your computer, you can try the code there as we go along.

```
# make sure we have a customerId from Step 2
if customerId:
    # build the request URL, using the actual userId in place of <userId>
    requestUrl = baseUrl + '/locations/userinfo/user/' + userId
    print('\nGetting **LOCATION** Information (' + requestUrl + ')\n');

    # create the request (not adding the 2nd data param means this is a GET request)
    request = urllib.request.Request(requestUrl)

    # add headers (for API authorization)
    for k, v in requestHeaders.items():
        request.add_header(k, v)

    # perform the request
    response = urllib.request.urlopen(request)
    results = response.read().decode(encoding)

    # create a dictionary from the results
    responseDict = json.loads(results)

    # print the results
    print(results)
else:
    print("error retrieving user information. 'customerId' was not present")
```

That's all we need to do for that API. We now have a list of domains available for this customer on this Smart+Connected Digital Platform instance.