Practical Exercise: Invoking the API

Training Objective

Learn how to test the API Developer Portal and build and deploy the web application and test the application using cURL.

Business Scenario

After subscribing to the API the partners can access the API through the web application which leverages the WSO2 API Manager token API to generate JWT access tokens on demand.

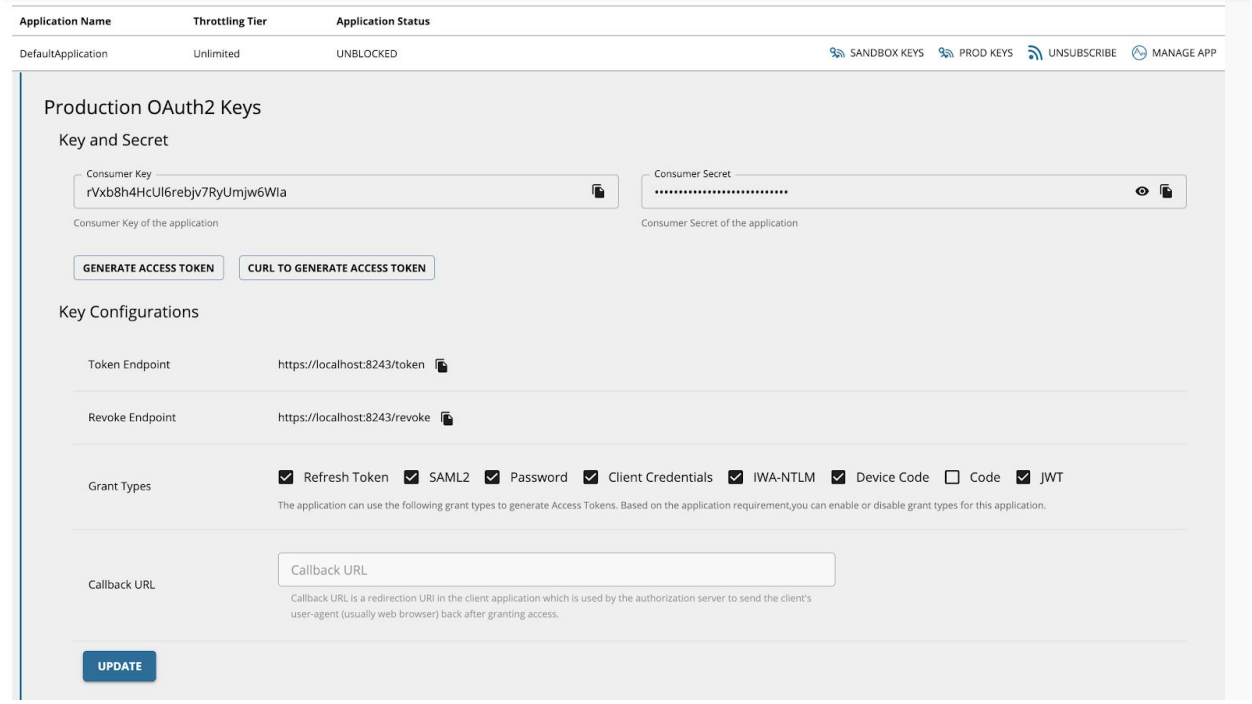
High-Level Steps

- Test the API using DevPortal
- Test the API using cURL
- Deploy and test the PizzaShack Application

Detailed Instructions

Test the PizzaShack Application from the DevPortal

1. Login to the DevPortal as the Subscriber user.
2. Click on the Pizzashack API.
1. Subscribe to the Pizzashack API by clicking on the **Subscriptions** tab of the API.
2. Once the subscription is added to the Subscriptions table click on the **PROD KEYS** tab and click on **Generate Keys**
3. Click on the **Generate Access Token** button and select the necessary scopes (order_pizza) and click on Generate and copy the generated Access Token.



4. Click on the **Try Out** tab.
5. Paste the access token in the access.token text box.
6. Expand the GET /menu resource and click on **Execute**. You will now be able to see the response of the GET /menu resource.

Test the API using cURL

To test the API through the API creator, we need to pass the right API key. The API Key must be passed inside an Authorization HTTP Header:

e.g.,
`Authorization: Bearer vMxNW6ILwNrWvnKJyewejSlHZFka`

It's very simple to use cURL - Let's exercise the Menu API.
Note: You can use a REST API Client of your preference

1. Open a new Command Line Interface
2. Type `curl -v http://localhost:8280/pizzashack/1.0.0/menu`

   OR
   Method: GET
   URL: `http://localhost:8280/pizzashack/1.0.0/menu`
   Authorization tab - Type : Bearer Token,
   Token : "d919d61a-8ef4-3059-b28e-9f38023aa306"

You will see the following message if the cURL command is used.

```
* About to connect() to localhost port 8280 (#0)
*   Trying 127.0.0.1... connected
.....
< HTTP/1.1 401 Unauthorized
< WWW-Authenticate: OAuth2 realm="WSO2 API Manager"
....
<ams:fault xmlns:ams="http://wso2.org/apimanager/security">
<ams:code>900902</ams:code>
<ams:message>Missing Credentials</ams:message>
<ams:description>
      Required OAuth credentials not provided
</ams:description>
</ams:fault>
```

3. Now use the **Access Token** you previously generated and add it as the Authorization Bearer. (Make sure you generate the access token with order_pizza scope)

   ```
   curl -H "Authorization: Bearer XXXXXXXX" -v http://localhost:8280/pizzashack/1.0.0/menu
   ```

   where XXXXXXXX is the access token generated through the application. You should get a response similar to the one below:

   ```
   * Adding handle: conn: 0x7fac09803a00
   * Adding handle: send: 0
   * Adding handle: recv: 0
   * Curl_addHandleToPipeline: length: 1
   ```

```
* - Conn 0 (0x7fac09803a00) send_pipe: 1, recv_pipe: 0
* About to connect() to localhost port 8280 (#0)
*   Trying ::1...
* Connected to localhost (::1) port 8280 (#0)
> GET /pizzashack/1.0.0/menu HTTP/1.1
> User-Agent: curl/7.30.0
> Host: localhost:8280
> Accept: */*
> Authorization: Bearer WnH0XfYEa0mInyMbnwhM0X24rKoa
>
< HTTP/1.1 200 OK
< Access-Control-Allow-Headers:
authorization,Access-Control-Allow-Origin,Content-Type
< Access-Control-Allow-Origin: *
< Access-Control-Allow-Methods: GET,PUT,POST,DELETE,OPTIONS
< Content-Type: application/json
< Date: Mon, 22 Dec 2014 05:41:09 GMT
* Server WSO2-PassThrough-HTTP is not blacklisted
< Server: WSO2-PassThrough-HTTP
< Transfer-Encoding: chunked
<
* Connection #0 to host localhost left intact
[{"name":"BBQ Chicken Bacon","description":"Grilled white chicken,
hickory-smoked bacon and fresh sliced onions in barbeque
sauce","icon":"/images/6.png","price":"14.99"},{"name":"Chicken
Parmesan","description":"Grilled chicken, fresh tomatoes, feta and
mozzarella
cheese","icon":"/images/1.png","price":"13.99"},{"name":"Chilly Chicken
Cordon Bleu","description":"Spinash Alfredo sauce topped with grilled
chicken, ham, onions and
mozzarella","icon":"/images/10.png","price":"21.99"},{"name":"Double
Bacon 6Cheese","description":"Hickory-smoked bacon, Julienne cut Canadian
bacon, Parmesan, mozzarella, Romano, Asiago and and Fontina
cheese","icon":"/images/9.png","price":"24.99"},{"name":"Garden
Fresh","description":"Slices onions and green peppers, gourmet mushrooms,
black olives and ripe Roma
tomatoes","icon":"/images/3.png","price":"12.99"},{"name":"Grilled
Chicken Club","description":"Grilled white chicken, hickory-smoked bacon
and fresh sliced onions topped with Roma
tomatoes","icon":"/images/8.png","price":"12.99"},{"name":"Hawaiian BBQ
Chicken","description":"Grilled white chicken, hickory-smoked bacon,
barbeque sauce topped with sweet
pine-apple","icon":"/images/7.png","price":"19.99"},{"name":"Spicy
Italian","description":"Pepperoni and a double portion of spicy Italian
sausage","icon":"/images/2.png","price":"27.99"},{"name":"Spinach
```

```
Alfredo","description":"Rich and creamy blend of spinach and garlic
Parmesan with Alfredo
sauce","icon":"/images/5.png","price":"17.99"},{"name":"Tuscan Six
Cheese","description":"Six cheese blend of mozzarella, Parmesan, Romano,
Asiago and Fontina","icon":"/images/4.png","price":"12.99"}]
```


Deploy and Test the PizzaShack Application

To test the application, log in to the API-M Management Console and do the following.

Note: Make sure the **am#sample#pizzashack#v1.war** file is already deployed under <APIM_HOME>/repository/deployment/server/webapps directory before starting.

7. Download the pizzashack.war file from the link and copy it to <APIM_HOME>/repository/deployment/server/webapps so that it is deployed.
8. The service will be deployed after a few seconds.

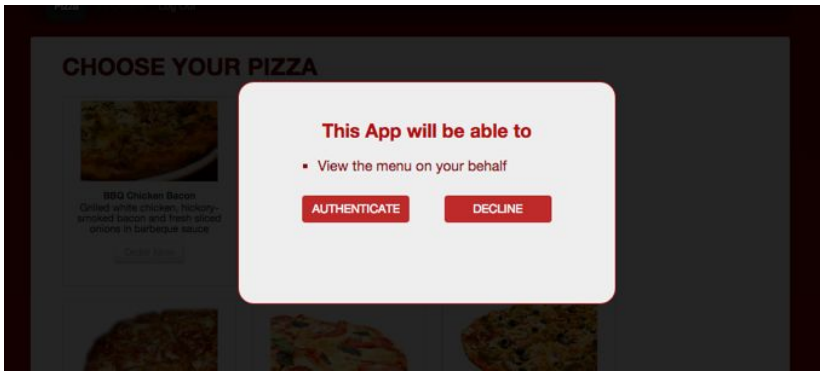
To edit the file and build the app.

3. Open <APIM_HOME>/repository/deployment/server/webapps/pizzashack/WEB-INF/web.xml.
4. Update the consumer key and client secret with the values obtained when generating the access token and save the file.Go to:

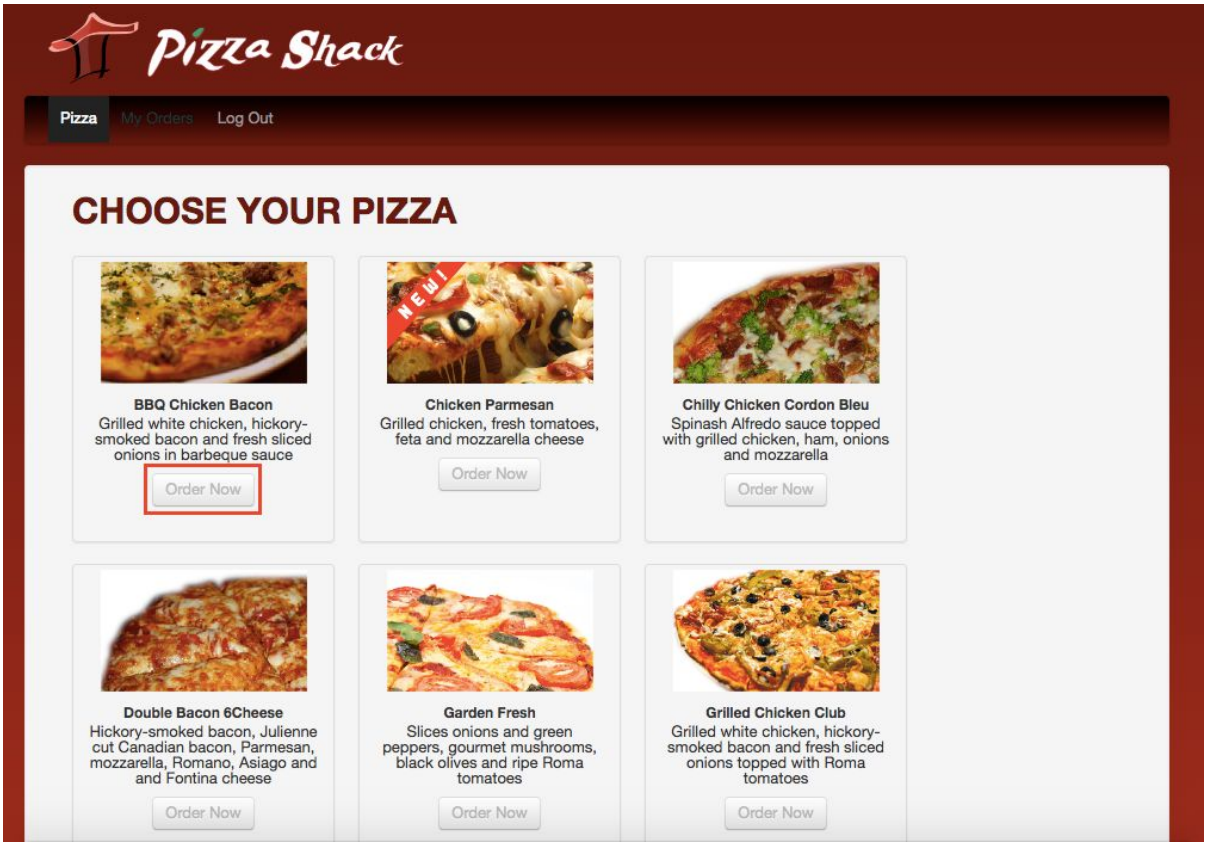
```
</context-param>
<context-param>
        <param-name>consumerKey</param-name>
        <param-value>XXXXXXXXXXXXXXXXXXXXXXXXXXXX</param-value>
</context-param>
<context-param>
        <param-name>consumerSecret</param-name>
        <param-value>YYYYYYYYYYYYYYYYYYYYYYYYYYYY</param-value>
 </context-param>
```

5. https://localhost:9443/pizzashack/login.jsp and log in as user mike.

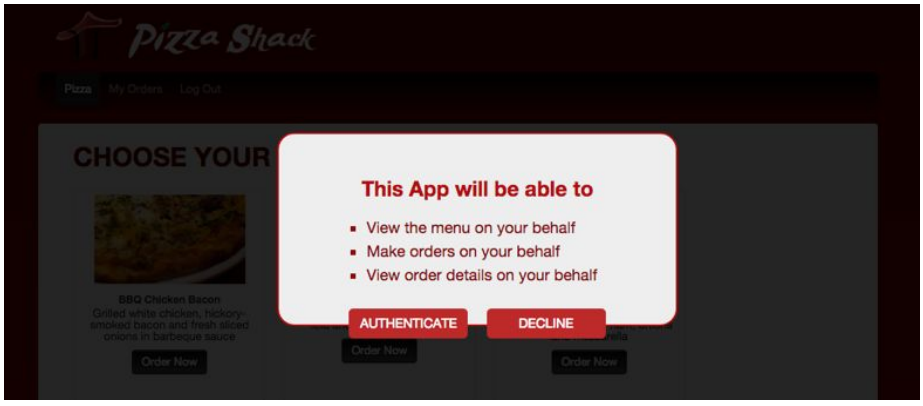
When mike logs in, he will not be able to get a token having the **order_pizza** scope since he doesn't have the **webuser** role. As a result, you will see the screen below.



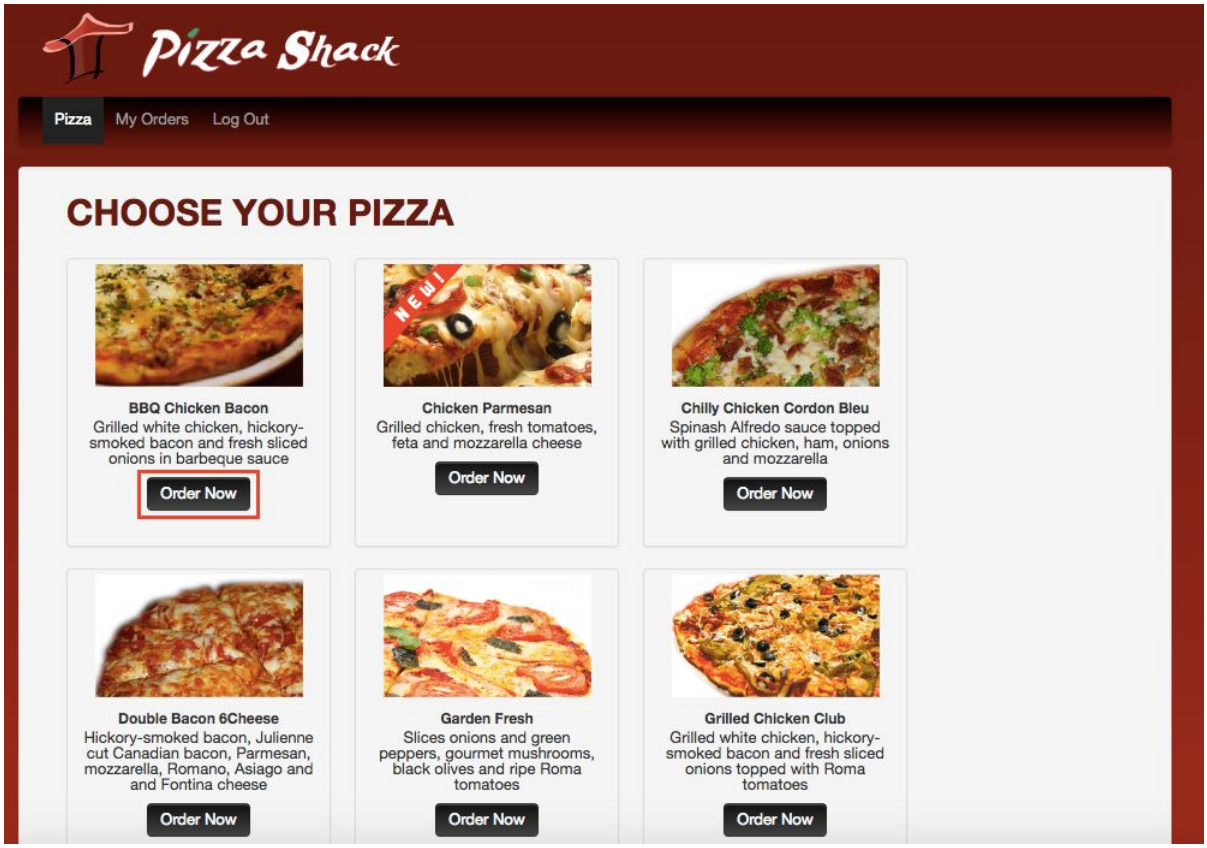
Note that the **Order Now** button is disabled.



6. Log in as a subscriber user. Since the user *subscriber* has the **internal/subscriber** role, he is capable of getting an access token which has the **order_pizza** scope and can invoke the /order resource of the PizzaShackAPI. When you log in as a subscriber, you will see the screen below.



Note that the **Order Now** button is enabled.



Expected Outcome

In this exercise, the API was tested using API Developer Portal and cURL and the PizzaShack web application was built and deployed in WSO2 API Manager. The web application was tested using 2 users with different levels of permission.