WSO2 API Manager 2.6.0

Labkit

Developer Fundamentals

training@wso2.com

WSO2 Inc. 787 Castro Street, Mountain View, CA 94041, USA

Tel: +1 408 754 7388 | **Fax:** +1 408 689 4328 | **Email:** training@wso2.com

Table of Contents

Lab: Getting Started with WSO2 API Manager

Training Objective

Verify that the products required for running this tutorial are installed and configured. and deploy and test the data required to work with the sample.

Note: The participants are expected to be connected to the internet throughout in order to successfully complete the lab exercises.

Business Scenario

PizzaShack Limited wants to extend their website for placing and managing online orders as a part of their effort in becoming the #1 online pizza shop. They have also found it increasingly useful to build an application for smartphones. The application is a Web application allowing you to choose and buy a Pizza online.They have subcontracted the development of the smartphone application to FunkyApps LLC. John Doe, Chief Architect of FunkyApps had some interesting feedback for PizzaShack. He suggested that the company considers monitoring of consumer statistics and probably looking into complex event processing in the future. John, also suggested that they make use of an API Store backed by a modern API Gateway providing security features such as OAuth 2.0 access tokens.

In order to achieve this, PizzaShack will be implementing WSO2 API Manager and a number of other WSO2 products for monitoring statistics, single sign on and so on.

The application leverages an API with 3 resources, which are exposed via the API Manager. Corresponding services are hosted in the WSO2 API Manager. WSO2 API-M Analytics Server will be used for monitoring.

High Level Steps

- Install WSO2 API Manager
- Install WSO2 API Manager Analytics
- Other installations
- Overview of the key directories in WSO2 API Manager
- Key configuration files
- Configure port offsets

Detailed Instructions

Install WSO2 API Manager

Before installing the product, ensure that the installation prerequisites have been met. Refer to the documentation [1] for detailed instructions. If prerequisites are fulfilled, instructions on installing the product can be found for:

- Linux or OS X at [2]
- Windows [3]

[1] Installation Prerequisites
[2] Installing on Linux
[3] Installing on Windows

Install WSO2 API Manager Analytics

Before installing the product, ensure that the installation prerequisites have been met. Refer to the documentation [1] for detailed instructions.

[1] Working with Analytics

Other Installations

In order to complete the use case described in this labkit the following products must be installed: A Rest API client or cURL [1], CLI tool [2] (Dev-Ops Tooling)

[1] https://www.getpostman.com/apps
[2] https://wso2.com/api-management/tooling/

For MAC OS X

Installing brew[1], JDBC driver for MySQL[2] and cURL[3]

[1] Install brew from https://brew.sh/
[2] In terminal run

- `brew tap gbeine/homebrew-java`
- `brew install mysql-connector-java`

[3] In terminal run `'brew install curl'`

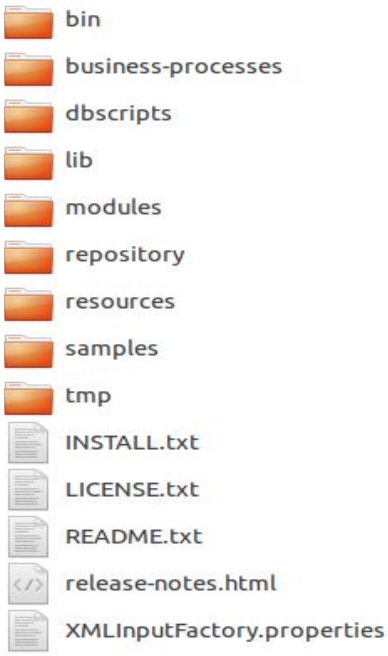
For Windows

Installing JDBC Drivers for SQL[1]

[1] Install Drivers from this location
[2] Install cURL from this location

Overview of the Key Directories in WSO2 API Manager

The structure of the <APIM_HOME> folder is as follows.



bin - This folder contains all the executable files including those scripts that are used to start/stop the application on Linux and Windows environments e.g.,:wso2server.sh and wso2server.bat.

business-processes – This folder contains information related to business process execution for API Management related operations. With WSO2 API Manager we have added 4 workflow plug points for the below operations: user creation process, application creation process, application registration process, subscription process.

dbscripts – A collection of database scripts required to create the Carbon database and the API Manager specific database on a variety of database management systems.

lib –The lib directory houses all the jar files that will be converted to OSGi bundles at startup and copied to the dropins directory.

modules – All the host objects belonging to the Jaggery module are declared within the modules folder in a file called module.xml.

repository – The main repository for all kinds of deployments and configurations in Carbon. This includes all default services, created APIs, Carbon configurations etc.

resources – Contains additional resources that may be used by the API-M.

samples - Sample APIs that can be used to explore the WSO2 API Manager functionality.

tmp – Will contain temporary files that are created when a product is run. These files will be cleared from time to time based on housekeeping tasks.

Key Configuration Files

File	Description
<PRODUCT_HOME>/repository/conf/carbon.xml	The carbon server configuration file
<PRODUCT_HOME>/repository/conf/api-manager.xml	The main configuration file that governs the API-M specific functionality.
<PRODUCT_HOME>/repository/conf/datasources/master_datasources.xml	The main configuration file for carbon datasources. Registry and User Manager refer the datasource configuration defined in this file.
<PRODUCT_HOME>/repository/conf/identity/identity.xml	The configuration file that governs identity related functionality.
<PRODUCT_HOME>/repository/conf/log4j.properties	The log4j configuration file used by WSO2 Carbon.
<PRODUCT_HOME>/repository/conf/registry.xml	The carbon registry configuration file. This will be used when the WSO2 Embedded Registry is used.
<PRODUCT_HOME>/repository/conf/user-mgt.xml	The User Manager configuration file used for configuring user management details.
<PRODUCT_HOME>/repository/conf/security/secret-conf.properties	The secret manager configuration that is used by the secret vault component.

Configure Port Offset

When you run multiple WSO2 products, multiple instances of the same product, or multiple WSO2 product clusters on the same server or virtual machines (VMs), you must change their default ports with an offset value to avoid port conflicts. The default HTTP and HTTPS ports (without offset) of a WSO2 product are 9763 and 9443 respectively. Port offset defines the number by which all ports defined in the runtime such as the HTTP/S ports will be changed. For example, if the default HTTP port is 9763 and the port offset is 1, the effective HTTP port will change to 9764. For each additional WSO2 product instance, you set the port offset to a unique value. The default port offset is 0.

There are two ways to set an offset to a port:

- Pass the port offset to the server during startup. The following command starts the server with the default port incremented by 3:. /wso2server.sh -DportOffset=3
- Set the Ports section of <PRODUCT_HOME>/repository/conf/carbon.xml as follows:<Offset>3</Offset>

We will be using API-M and APIM Analytics for these exercises. Since both these servers need to run in the same machine for this demo, we must change the port offset in home/repository/conf/carbon.xml file. Enter the following port offsets for each product:

File	Port Offset
<API-M_HOME>/repository/conf/carbon.xml	0
<Analytics_HOME>/repository/conf/worker/deployment.yaml	1

Expected Outcome

As the API-M was not given an offset, it will run on the default port while the other products will run on the relevant according to the given port offset.

[1] Running the Product - Docs

Lab: Defining Users and Roles

Training Objective

In this section, you will learn how to set up custom roles and users. Roles contain permissions for users to manage the server. You can create different roles with various combinations of permissions and assign them to a user or a group of users. User roles can be reused throughout the system and prevent the overhead of granting multiple permissions to each and every user individually.

Business Scenario

PizzaShack has an employee who will be creating the menu, order and delivery APIs and another employee who will be publishing this to the website. API consumers can log in to the site and access these APIs. Separate user roles and users should be created for API creators and publishers.

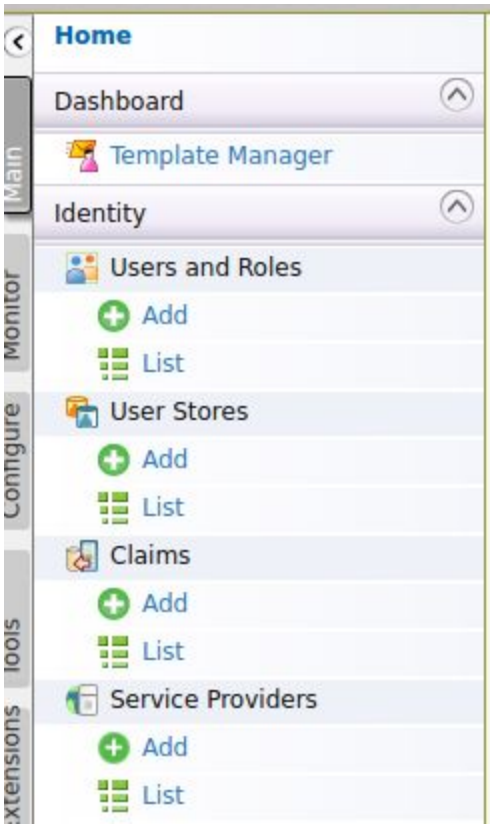
High Level Steps

- Define roles
- Define users via the admin console

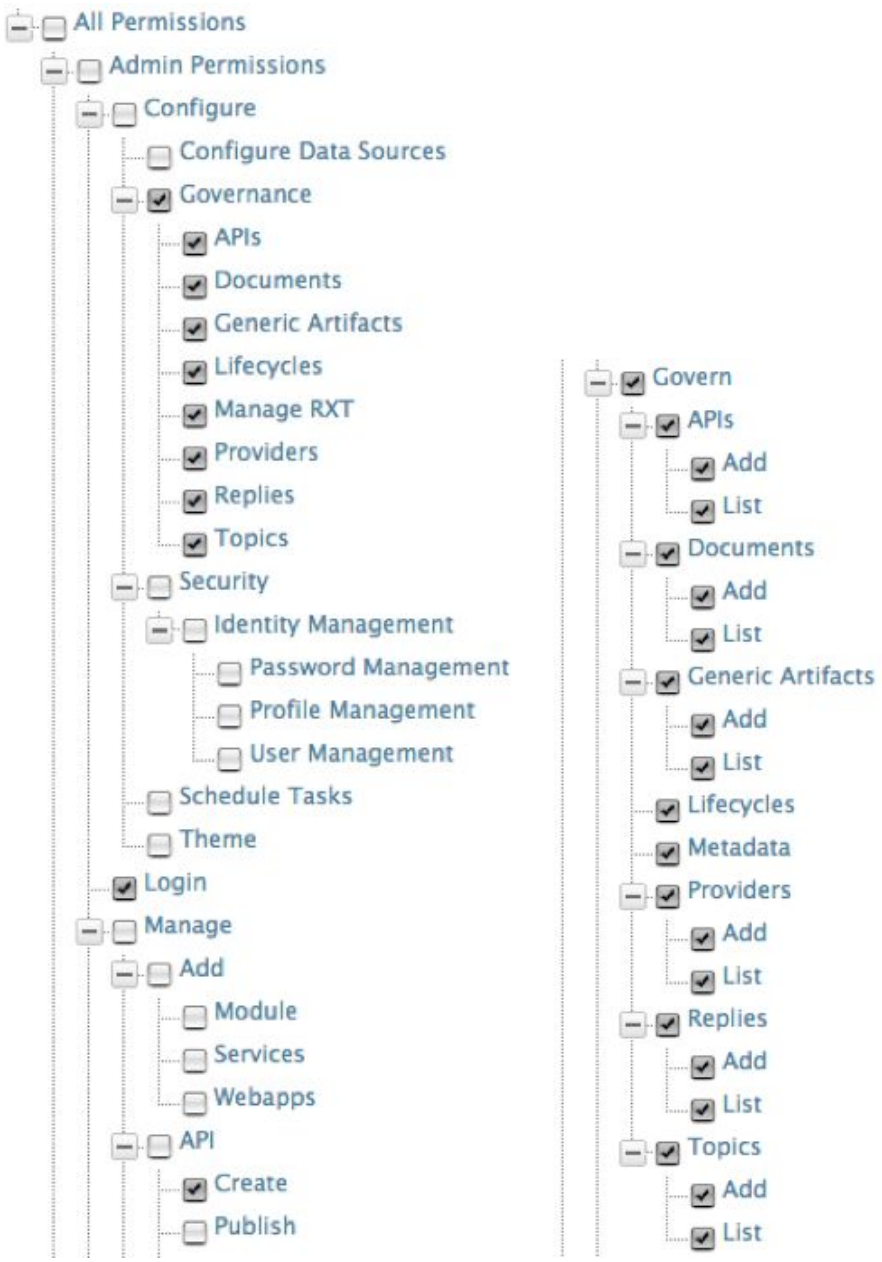
Detailed Instructions

Define Roles

1. Open a Command Line Interface.
2. Start the WSO2 API Manager by navigating to the <API-M_HOME>/bin directory and running wso2server.bat (on Windows) or sh wso2server.sh (on Linux)
3. Log in to the API-M admin console, which is available by default at: https://localhost:9443/carbon. You can log in to the console using the default admin/admin credentials.



4. Click **Main** > **Users and Roles**> **Add > Add New Role**.
5. Provide **apicreator** as the role name.
6. Click **Next** - You will be presented with a list of permissions. For the creator role, you need to select the following permissions:
   - **Configure** > **Governance** and all underlying permission
   - **Login**
   - **Manage** > **API** > **Create**
   - **Manage** > **Resources** > **Govern** and all underlying permissions.



7. Click **Finish** (at the bottom of the page).

Repeat the steps to create the **apipublisher** role, with the **following permissions**:

- **Login**
- **Manage** > **API** > **Publish**
  Finish

Define Users via the Admin Console

You can now create a user in each of those roles. To do so:

1. Click **Main** > **Users and Roles**> **Add** > **Add New User**
2. Provide user name (apicreator) and password (password)
3. Click **Next**.
4. Select the **apicreator** role.
5. Click **Finish**.

Note : You can also choose the **creator** role available by default in the management console, in Step 5

Repeat the steps to create a user (apipublisher) in the **apipublisher** role.
Note : You can also choose the **publisher** role available by default in the management console.

Create a role named 'webuser' and grant it **Login** permission:

1. Click **Main** > **Users and Roles** > **Add**.
2. Click **Add New Role**.
3. Provide **webuser** as the role name.
4. Click **Next** - You will be presented with a list of permissions. Select the **Login** permission.
5. Click **Finish**.

Create a user named 'john' and assign him the 'webuser' role:

1. Click **Main** > **Users and Roles** > **Add**.
2. Click **Add New User**.
3. Provide user name (john) and password (password)
4. Click **Next**.
5. Select the **'webuser'** role.
6. Click **Finish**.

Create a user named 'mike'. Do not assign him any roles:

1. Click **Main** > **Users and Roles** > **Add**.
2. Click **Add New User**.
3. Provide user name (mike) and password (password)
4. Click **Finish**.

Expected Outcome

A creator role with permissions for creating APIs and a publisher role with permissions for publishing APIs have been created. Users named 'apicreator' and 'apipublisher' have been created and given the appropriate roles. A role named 'webuser' and users named 'john' and 'mike' have been created to test out the PizzaShack application.

Creating and Publishing the PizzaShackAPI

Training Objective

Learn how to create an API, add documentation to it and publish it to the store using the Publisher.

Business Scenario

After setting up the API-M, the API is created and published through the API Publisher in order to make it subscribable from the store.

Business Scenario: PizzaShack Limited is providing a store from which consumers can subscribe to their API This works as a secondary business function for PizzaShack and attracts many developers to the PIzzaShack website. The API will be comprehensively documented for ease of use.

High Level Steps

- Add the PizzaShackAPI to the store
- Implement APIs
- Manage APIs
- Add documentation
- Publish the APIs

Detailed Instructions

Adding the PizzaShackAPI to the Store

Now that we set up the API-M and added users, we are ready to publish the API the PizzaShack application requires.

To add the API to the store, follow those steps:

1. Open the API Publisher web application from https://localhost:9443/publisher.
2. Log in using the user in creator role you defined previously (apicreator).
3. Click **Add New API**.
4. Select **Design a New Rest API**.
5. Click **Start Creating**.
6. Provide information on the API as per the table below

Field	Value	Description
Name	**PizzaShackAPI**	Name of API as you want it to appear in the API store
Context	**pizzashack**	URI context path that is used by API consumers (Application Developers)
Version	**1.0.0**	API version (in the form of version.major.minor)
Access Control	All	The ability to protect an API to be managed only by users with specific roles
Visibility	Public	Whether this API is visible to all or restricted to certain roles.
Thumbnail Image	Download a PizzaShack logo image and upload it Get the Logo here : Link	Icon to be displayed in API store (can be jpeg, tiff, png format) - Under **Advanced Options**.
Description	PizzaShackAPI: Allows to manage pizza orders (create, update, retrieve orders)	High level description of API functionality
Tags	pizza, order, pizza-menu	One of more tags. Tags are used to group/search for APIs (Press **Enter** after each tag)
API Definition		An API is made up of one or more resources. Each resource handles a particular type of requests. A resource is analogous to a method (function) in a larger API. API resources can accept following optional attributes: ● **verbs**: Specifies the HTTP verbs a particular resource would accept. Allowed values are GET, POST, PUT, DELETE. Multiple values can be specified. ● **uri-template**: A URI template as defined in http://tools.ietf.org/html/rfc6570 (eg: /phoneverify/{phoneNumber}) ● **url-mapping**: A URL mapping as defined as per the servlet specification (extension mappings, path mappings and exact mappings)

For the PizzaShackAPI, we will be defining 4 resources as defined below.

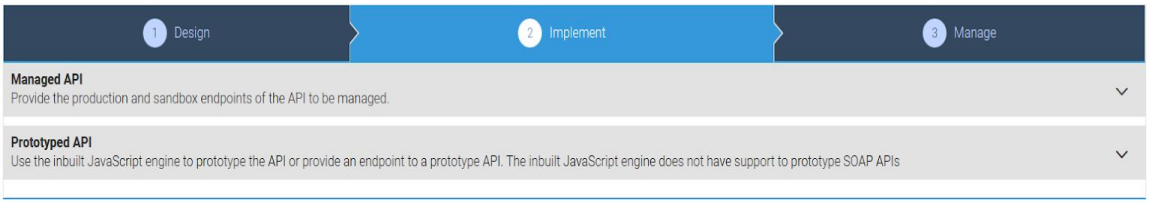
7. Enter the Resource URL and click **Add** and repeat for each **Resource URL**.

Resource URL	Methods
menu	GET
order	POST
order/{orderid}	GET
order/{orderid}	PUT

8. Click I**mplement**.

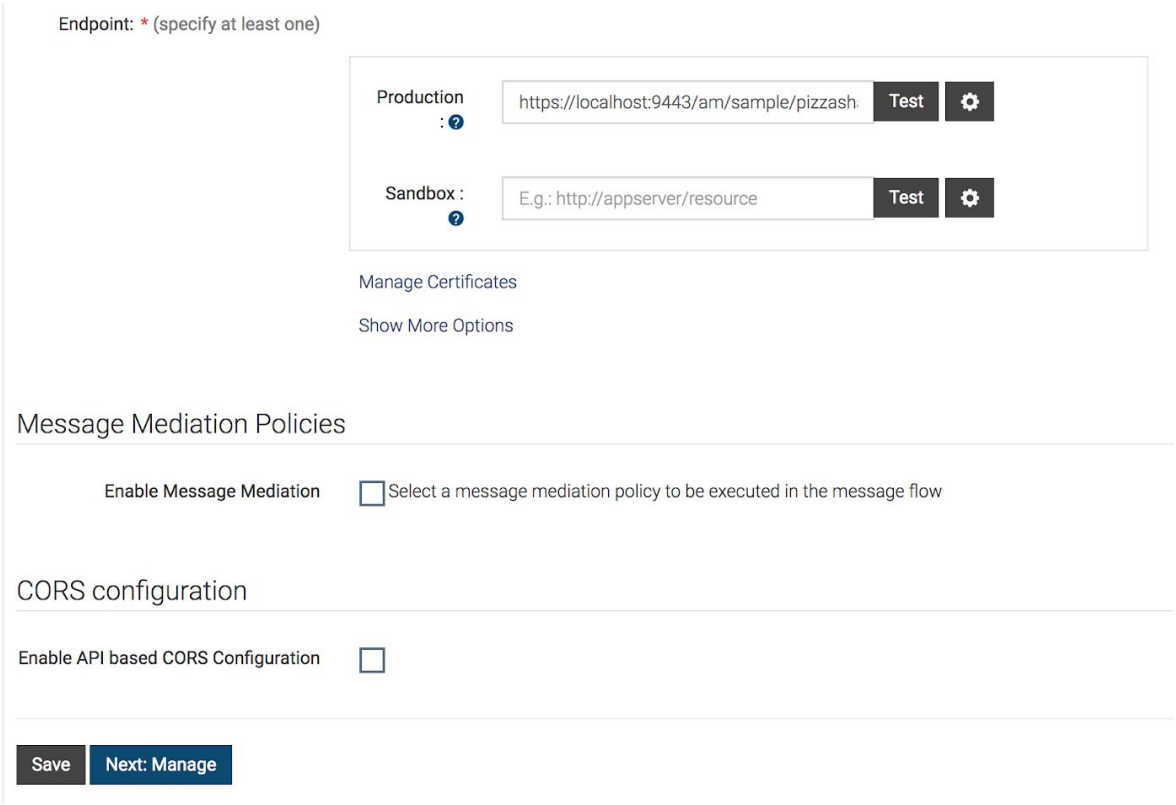
Implement APIs

1. Select **Managed API**.

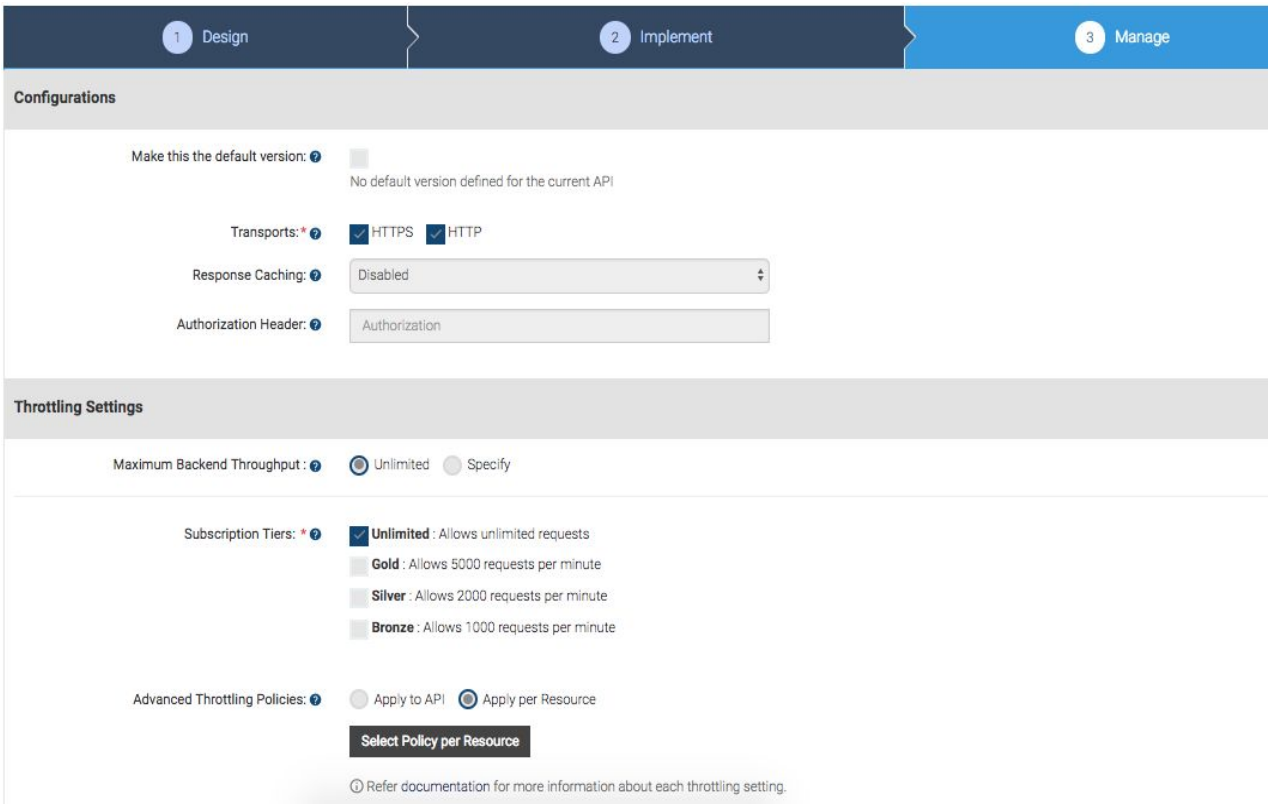


2. Specify the following.

Field	Value	Description
Endpoint Type	HTTP/REST Endpoint	
Production Endpoint	https://localhost:9443/am/sample/pizzashack/v1/api/	Endpoint of the backend service URL
Sandbox URL	N/A	Endpoint of sandbox (testing) back end service. A sandbox URL is meant to be used for online testing of an API with easy access to an API key. We have no sandbox in this sample.
SELECT "Show more options"		
Endpoint Security Scheme:*	Non Secured	
Enable Message Mediation	Not Selected	Define your own message mediation policy for incoming and outgoing messages.
Enable API based CORS Configuration	Not Selected	Enable CORS for the API



3. Click Next: Manage. Which will take you to the next page



Manage APIs

Managing an API involves specifying its management attributes such as throttling tiers, external sequences, and so on. Provide the following information on the **Manage** tab of the API.

Field	Value	Description
Make this default version	Not selected	The default version option allows you to mark one API, from a group of API versions, as the default one, so that it can be invoked without specifying the version number in the URL.
Transports	HTTP/HTTPS	APIs can be exposed in HTTP and/or HTTPS transport: The transport protocol on which the API is exposed. Both HTTP and HTTPS transports are selected by default. If you want to limit API availability to only one transport (e.g., HTTPS), un-check the other transport.
Response Caching	Disabled	Response caching is used to enable caching of response messages per API. Caching protects the backend systems from being exhausted due to serving the same response (for same request) multiple times. If you select the enable option, specify the cache timeout value (in seconds) within which the system tries to retrieve responses from the cache without going to the backend.
Maximum Backend Throughput	Unlimited	Limits the total number of calls the API Manager is allowed to make to the backend. While the other throttling levels define the quota the API invoker gets, they do not ensure that the backend is protected from overuse. Hard throttling limits the quota the backend can handle.

Subscription Tiers	Unlimited	The API can be available at different level of service; you can select multiple entries from the list. At subscription time, the consumer chooses which tier they are interested in.
Advanced Throttling Policies	Apply per Resource	Throttling policies can be applied per resource (different policies for each resource) or one policy for all resources

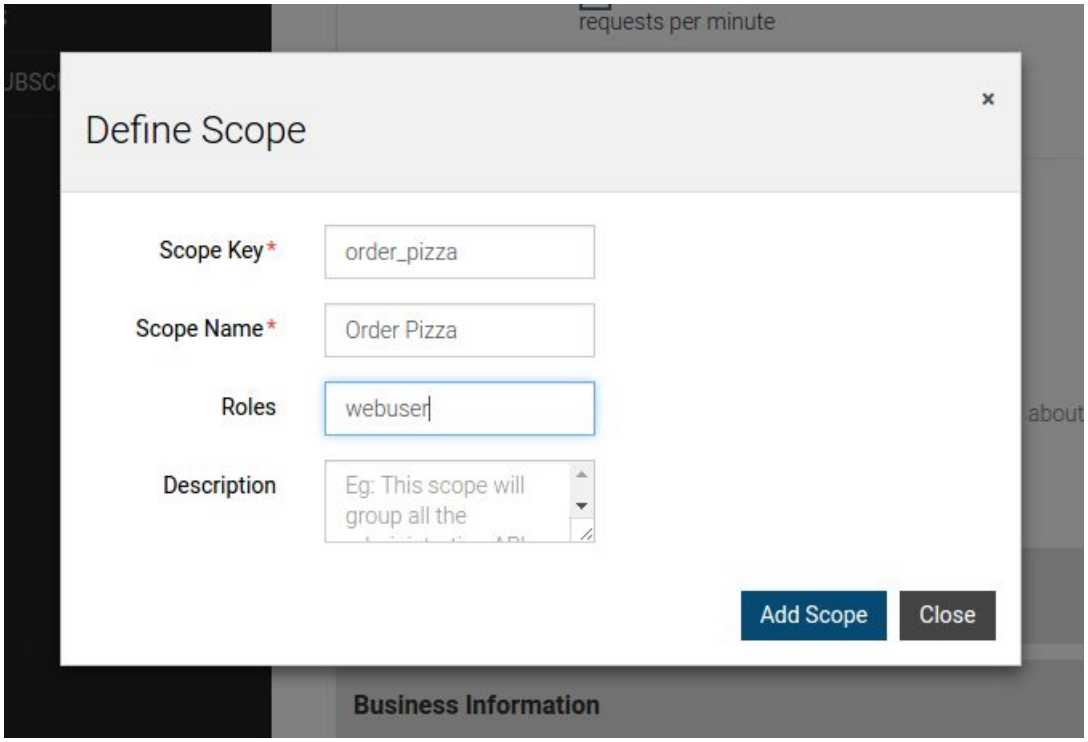
For this use case, we will also be making use of OAuth2.0 scopes. Therefore we will be creating a scope named order_pizza and allowing that scope to only users with webuser role.

Scopes enable fine-grained access control to API resources based on user roles. You define scopes to an API's resources. When a user invokes the API, his/her OAuth 2 bearer token cannot grant access to any API resource beyond its associated scopes.

Field	**Description**
Scope Key	A unique key for identifying the scope. Typically, it is prefixed by part of the API's name for uniqueness, but is not necessarily reader-friendly.
Scope Name	A human-readable name for the scope. It typically says what the scope does.
Roles	The user role(s) that are allowed to obtain a token against this scope. E.g., manager, employee.

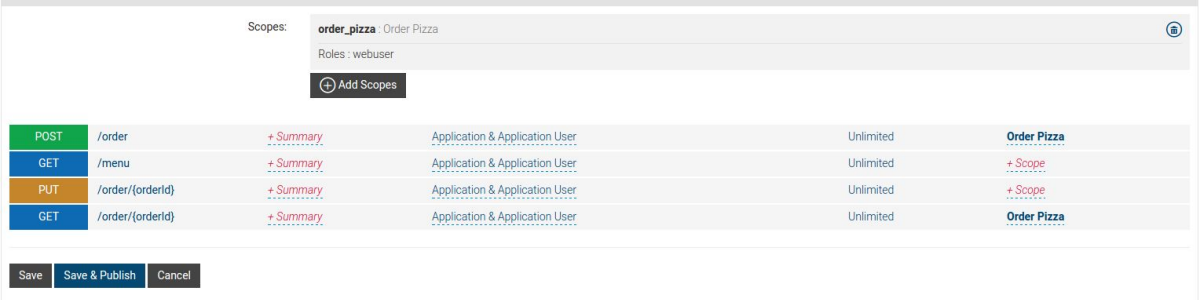
To invoke an API protected by scopes, you need to get an access token via the Token API. Tokens generated from the **APPLICATIONS** page in the API Store will not work.

1. Click **Add Scopes**.
2. Enter the following information and click **Add Scope**.



Field	Value
Scope Key	order_pizza
Scope Name	Order Pizza
Roles	webuser
Description	None

3. Once the scope is defined, we need to assign that scope to the appropriate resources.



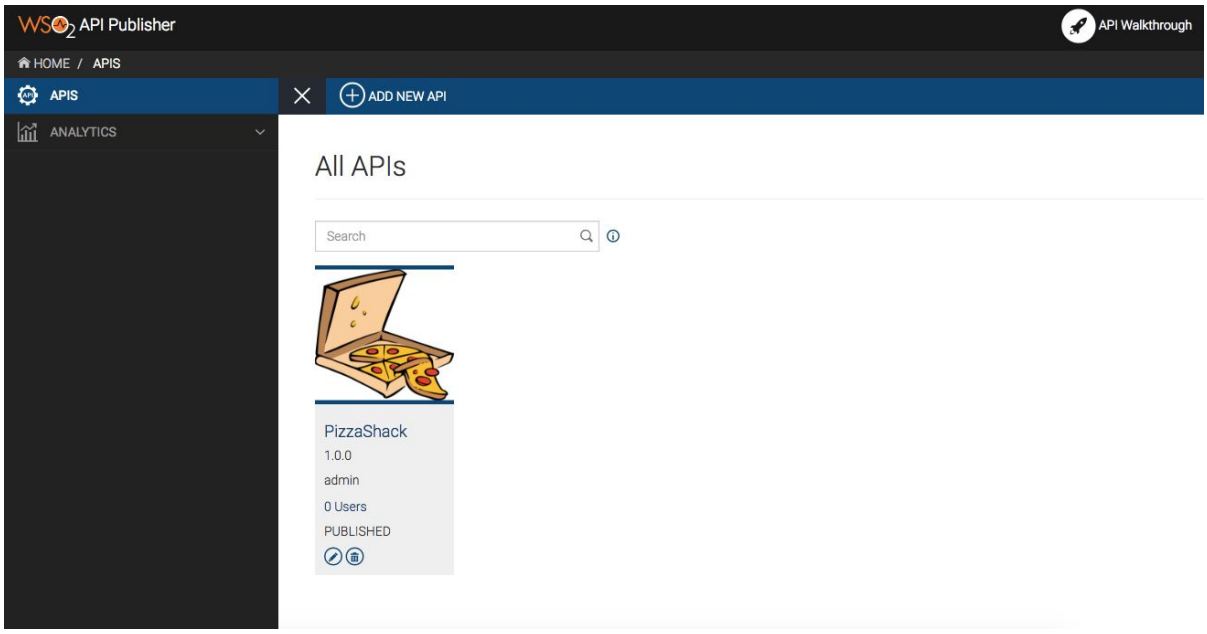
Note: The order in which the scopes are shown in the image above can differ from the order on screen. Make sure you add the scopes on the POST/order and GET/order{orderid}.

Once a request has been accepted by a resource, it will be mediated through an in-sequence. Any response from the back-end is handled through the out-sequence. A fault sequence is used to mediate any unhandled errors that might occur in either the in or out sequence. Default in-sequence, out-sequence and fault sequence are generated when the API is published.

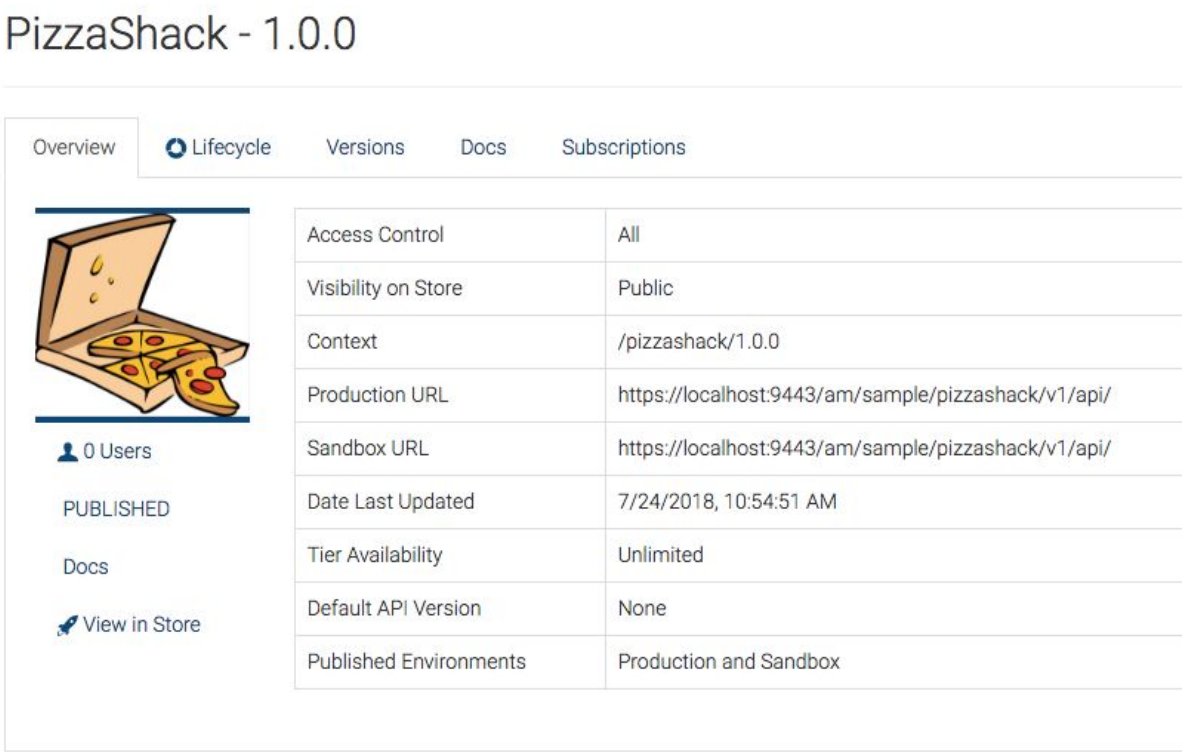
4. Click **Save**.
5. Click on “Go To Overview” once saved

Add Documentation

1. Once the API has been created, click **Browse** and then click the PizzaShack icon and open its details.



You see something similar to the image below:



2. Click the **Docs** tab and add documentation to the API. Documentation can be provided inline or via a URL or file. For inline documentation, you can edit the contents directly from the API publisher interface.

Several documents types are available:

- How To
- Samples and SDK
- Public Forum
- Support Forum
- Other

To create a How-To document:

3. Select the **How To** type.
4. Provide a name for the document, such as “How to use this API”.
5. In Summary, enter “Describe how to use this API”.
6. Provide a short description of the document (this will appear in the API store).
7. Choose the **Inline** option under **Source**.
8. Click **Add Document**.

Once the document has been added, you can edit the contents by clicking on the **Edit Content** link. An embedded editor allows you to edit the document contents.

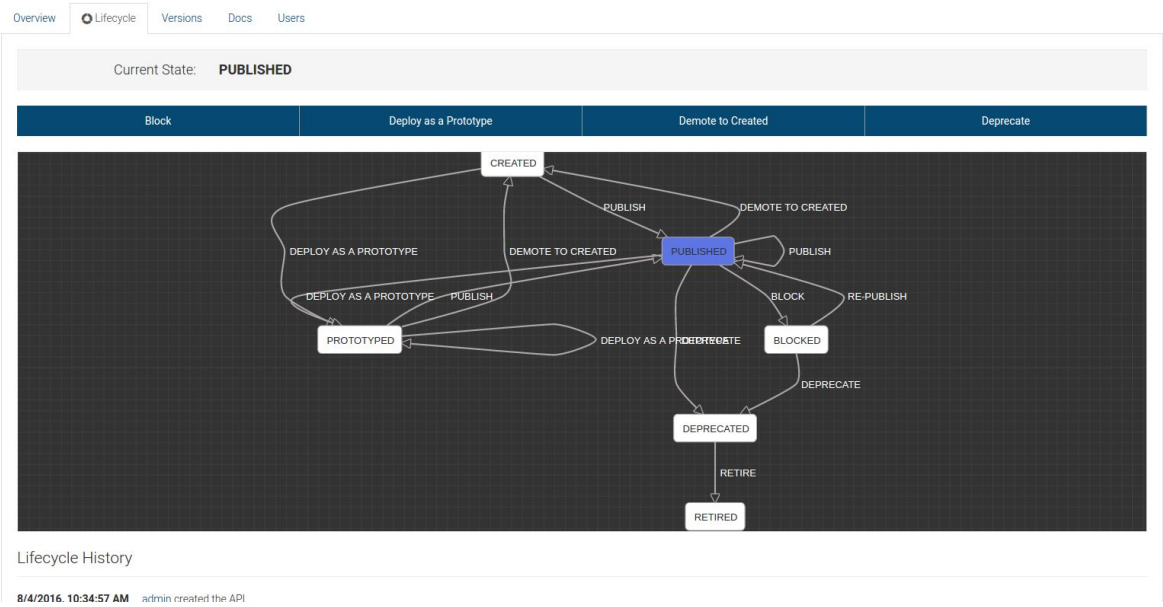
Publish the API

The API is now ready to be published. This has to be done by a user with the publisher role.

To publish the API:

1. Log out as apicreator and login as apipublisher.
2. Click on the API - You can see that an additional tab named **Lifecycle** is now available, allowing you to manage the API lifecycle.
3. To publish the API, select **PUBLISH**.

The API is now published and visible to consumers in the API store. The API life cycle history is visible at the bottom of the page.



Expected Outcome

The PizzaShackAPI which manages pizza orders has been created and published and can be accessed through the Store.

Lab : Import/Export API

Training Objective

In this section you will learn how to use the API Import Export tool. You will learn how to move an API artifact among different API Manager setups. Learn how to zip the PizzaShackAPI (created in previous lab exercises) to a zip file and try to import it again using the API Import Export Tool.

Business Scenario

PizzaShack maintains its APIs in multiple environments. The APIs created in one environment has to be moved to another completely. PizzaShack has also created an API to check the weather reports in an environment. The Weather API has to be imported to the production environment.

High Level Steps

- Set up API Import Export tool
- Run the CLI
- Export PizzaShackAPI as zip file
- Import API using the API Import Export Tool

Deploy the API Import/Export tool

1. Download the latest import/export tool from here.
2. Copy the downloaded war file to the <API-M_HOME>/repository/deployment/server/webapps folder.
3. Start API Manager. If the server is already started, the war file will be automatically deployed.

Run the CLI

1. Navigate to the location of the CLI tool through the command prompt.
2. Start the tool using `./apimcli` command.
3. Add the production environment using the following command.

```
./apimcli add-env -n production \
                    --registration
https://localhost:9443/client-registration/v0.14/register \
                    --apim https://localhost:9443 \
                    --token https://localhost:8243/token \
                    --import-export https://localhost:9443/api-import-export-2.6.0-v1 \
                    --admin https://localhost:9443/api/am/admin/v0.14 \
                    --api_list https://localhost:9443/api/am/publisher/v0.14/apis \
                    --app_list https://localhost:9443/api/am/store/v0.14/applications
```



```
~ $./apimcli add-env -n production \
>                       --registration https://localhost:9443/client-registration/v0.14/register \
>                       --apim https://localhost:9443 \
>                       --token https://localhost:8243/token \
>                       --import-export https://localhost:9443/api-import-export-2.6.0-v1 \
>                       --admin https://localhost:9443/api/am/admin/v0.14 \
>                       --api_list https://localhost:9443/api/am/publisher/v0.14/apis \
>                       --app_list https://localhost:9443/api/am/store/v0.14/applications
Successfully added environment 'production'
```

Export and Import an API

Exporting an API

1. Run the following command in the CLI to export the PizzaShackAPI as a zipped file.
   ./apimcli export-api -n <API-name> -v <version> -r <provider> -e <environment> -u <username> -p <password> -k

   Sample command :
   ./apimcli export-api -n PizzaShackAPI -v 1.0.0 -r apicreator -e production -u admin -p admin -k

```
apimcli $./apimcli export-api -n PizzaShackAPI -v 1.0.0 -r apicreator -e production -u admin -p admin -k
Succesfully exported API!
Find the exported API at /Users/[redacted]/.wso2apimcli/exported/apis/production/PizzaShackAPI_1.0.0.zip
apimcli $cd
~ $cd /Users/[redacted]/.wso2apimcli/exported/apis
apis $ls
production
apis $cd production
production $ls
PizzaShackAPI_1.0.0.zip
```

Note: -k is used to accept self signed certificate

Importing an API

1. Download the WeatherAPI.zip from here.
2. Move it to this location /Users/john/.wso2apimcli/exported/apis/WeatherAPI.zip (your account user name should be replaced with john).
3. Run the following command in the CLI to import the API.
   ./apimcli import-api -n <API-name> -e <environment> -f <ExportedPack.zip> -u <username> -p <password> -k

   Sample command :
   ./apimcli import-api -f WeatherAPI.zip -e production -u admin -p admin -k

```
apimcli $./apimcli import-api -f WeatherAPI.zip -e production -u admin -p admin -k
ZipFilePath: /Users/[redacted]/.wso2apimcli/exported/apis/WeatherAPI.zip
Successfully imported API 'WeatherAPI.zip'
Succesfully imported API!
apimcli $
```

4. The weather api should now be visible after logging in to the portal.

Lab: Working with Tenants

Training Objective

Learn how to create tenants and use tenants to share APIs.

Business Scenario

PizzaShack Limited would like to create a separate tenant for employees in order to share APIs only with them. The first API that will be shared will be used for capturing statistics on customers. This API will be shared by the PizzaShack head office and shared among the branches.

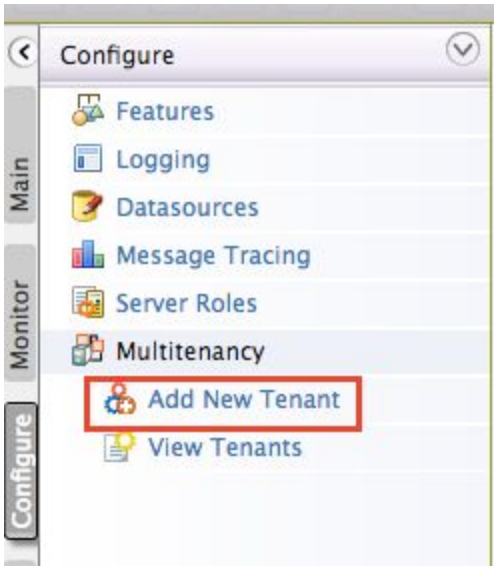
High Level Steps

- Create tenants
- Share API within tenant

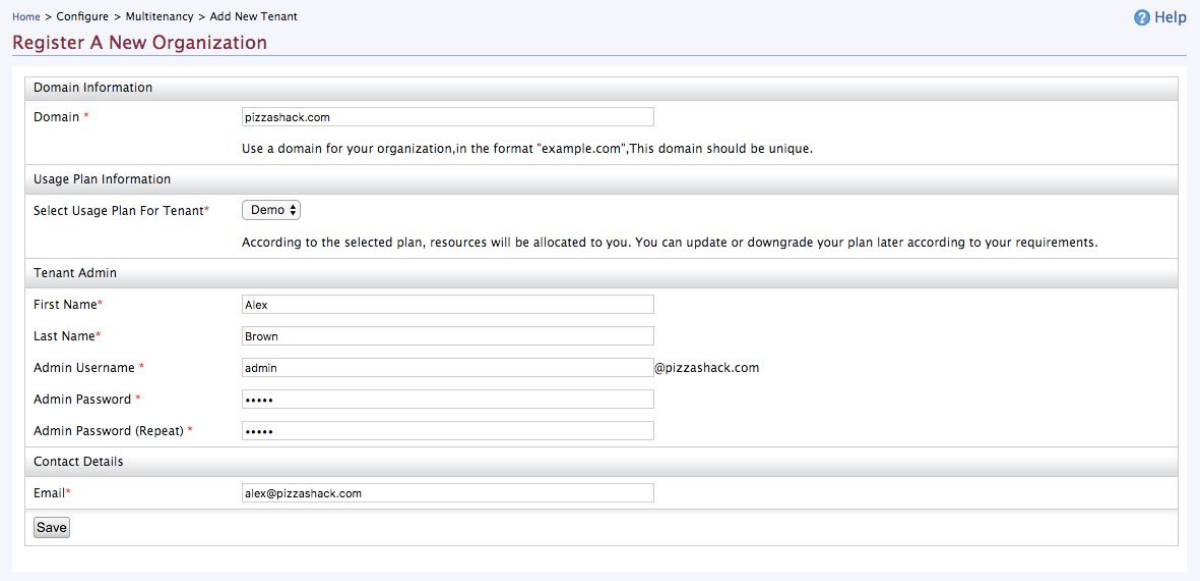
Detailed Instructions

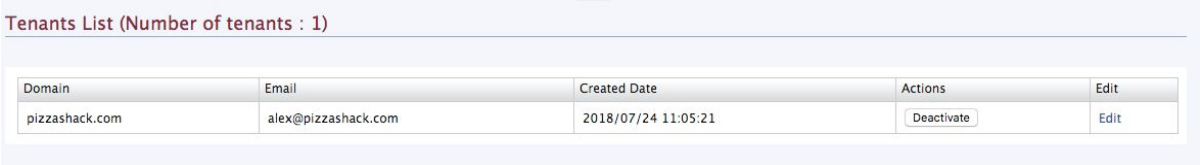
Create Tenants

1. Log in to the Management Console as an admin user.
2. Click **Configure** > **Multitenancy** > **Add New Tenant**.

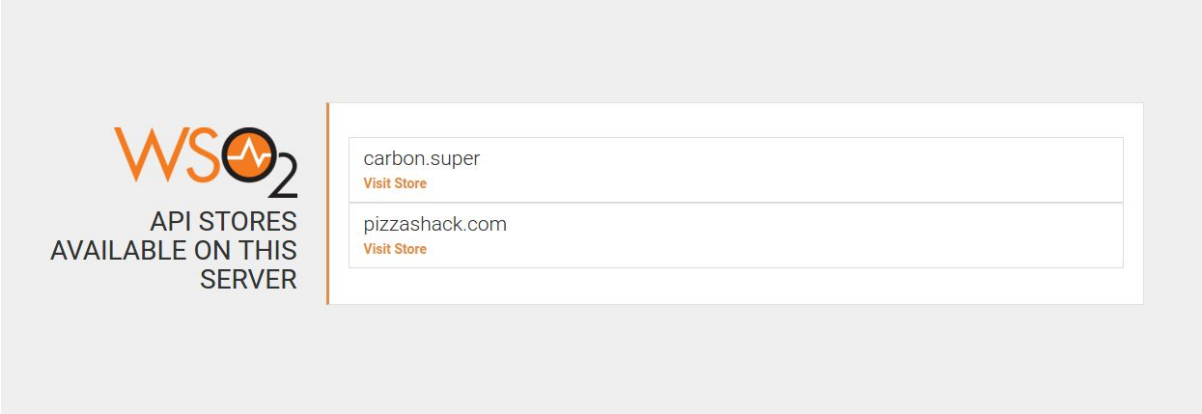


3. Register a new domain named pizzashack.com as follows:

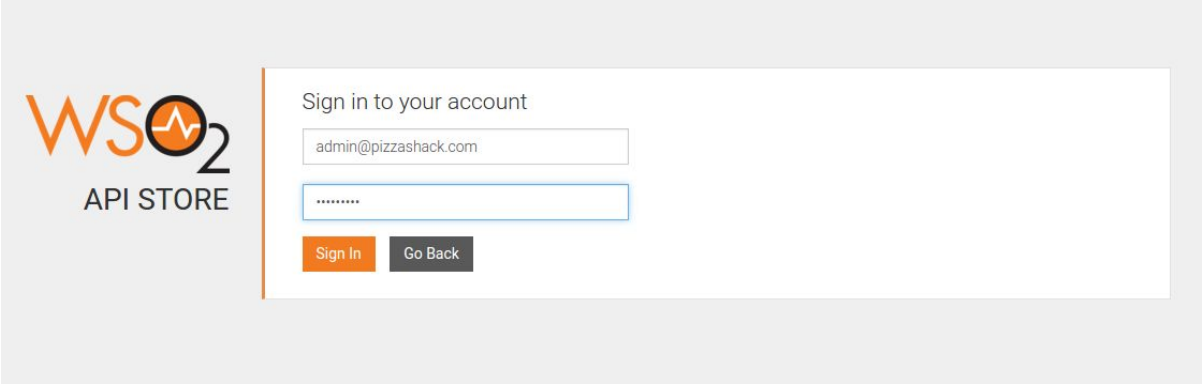




4. Go to the URL for the store https://localhost:9443/store/. The super tenant and the newly created tenant will be displayed.

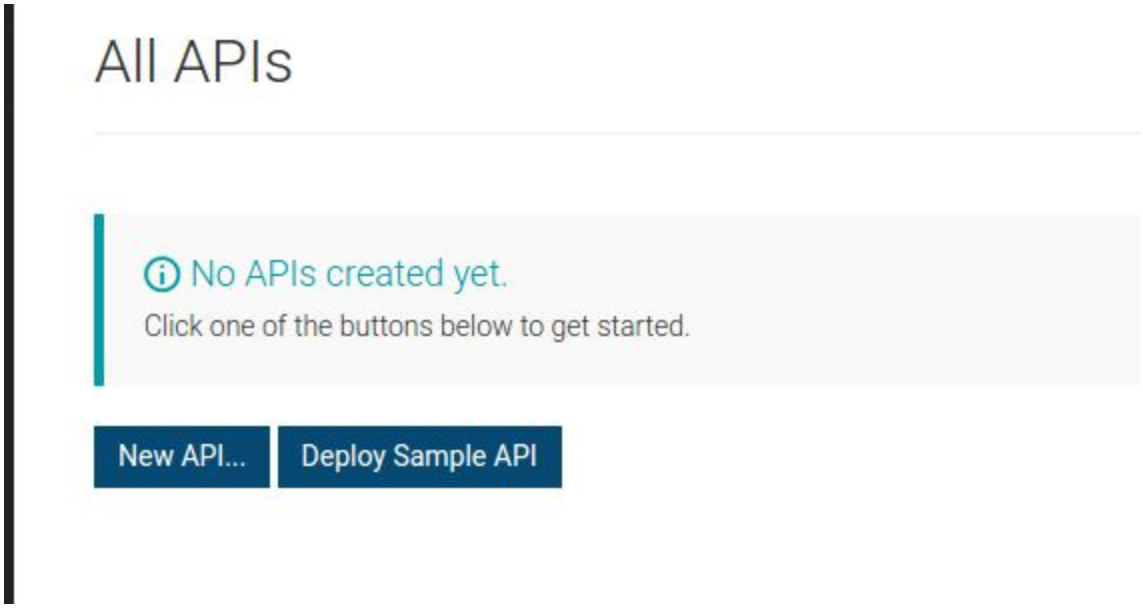


5. Click on pizzashack.com and note that no APIs have been published yet.
6. Login as admin@pizzashack.com. There are no APIs displayed even after logging in.

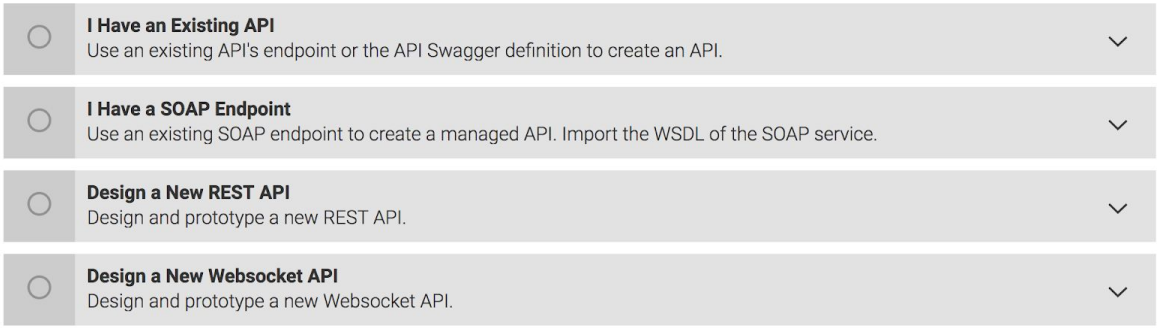


Manage the API within Tenant

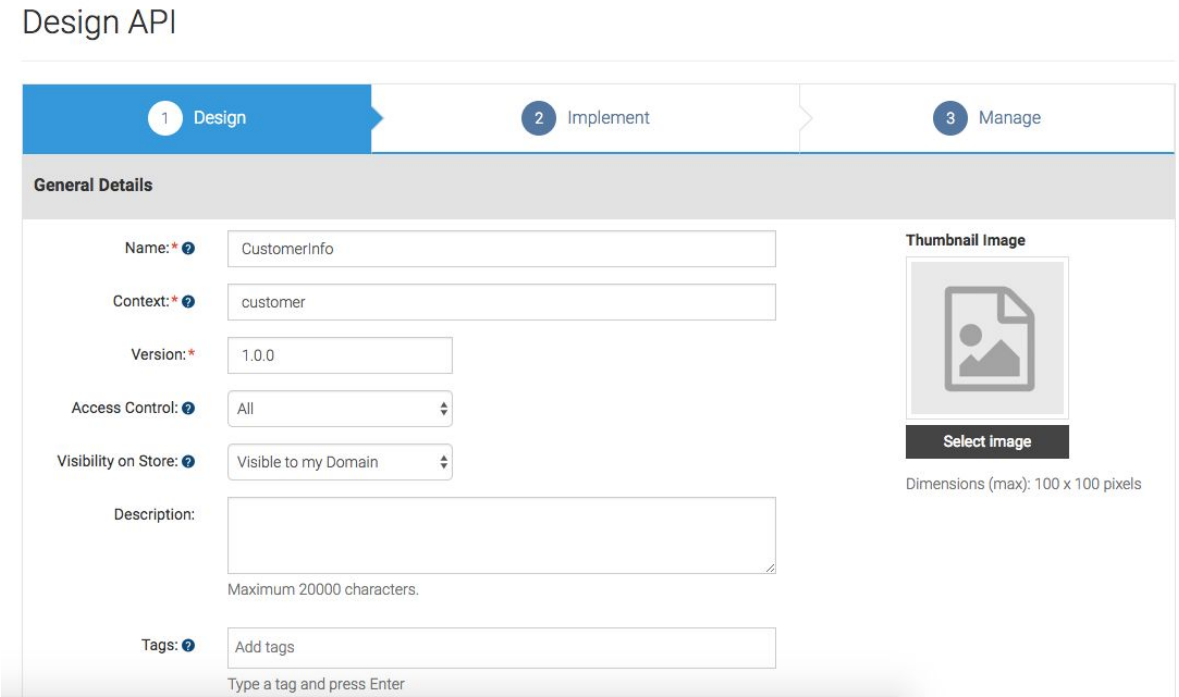
1. Log in to the Publisher as admin@pizzashack.com.
2. Click **New API...**.



3. Click **Design a new REST API** and then click **Start Creating**.

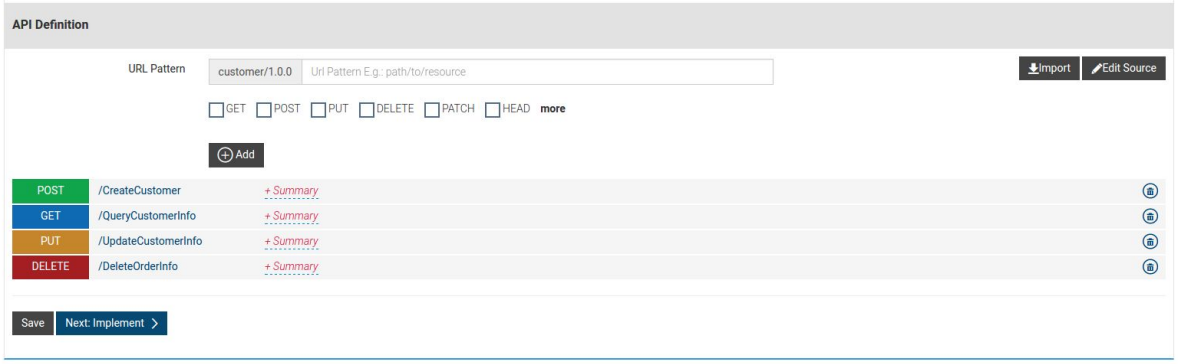


4. Enter the following details.



5. Add the following resources.

Resource URL	Methods
CreateCustomer	POST
QueryCustomerInfo	GET
UpdateCustomerInfo	PUT
DeleteOrderInfo	DELETE



6. For the **CreateCustomer** resource, change **Required** to **True** for the **Payload** parameter.

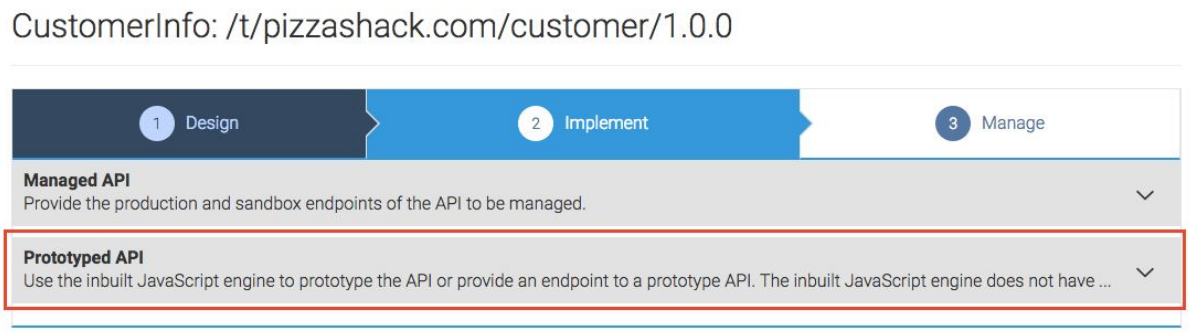
6. Add the **MobileNumber** parameter for the **QueryCustomerInfo** resource:

7. For the **UpdateCustomerInfo** resource, change **Required** to **True** for the **Payload** parameter.

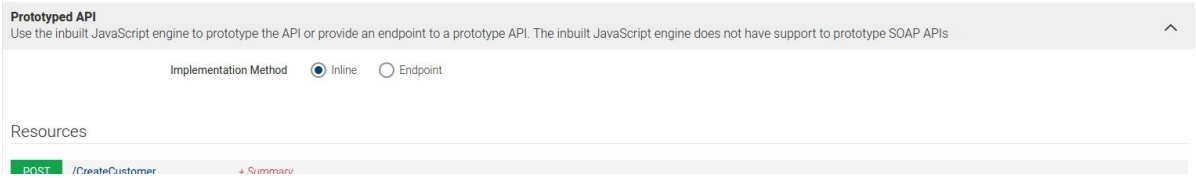
8. Add the **orderid** parameter to the **DeleteOrderInfo** resource.

9. Click **Implement**.

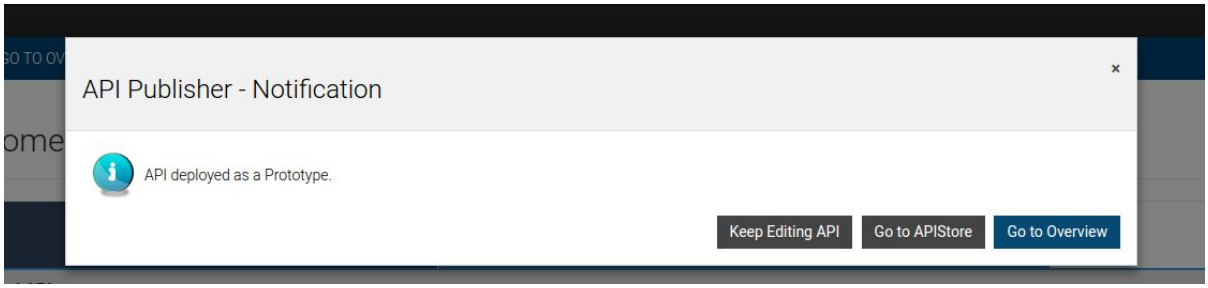
10. Click **Prototyped API**.



11. Select **Inline** as the implementation method.

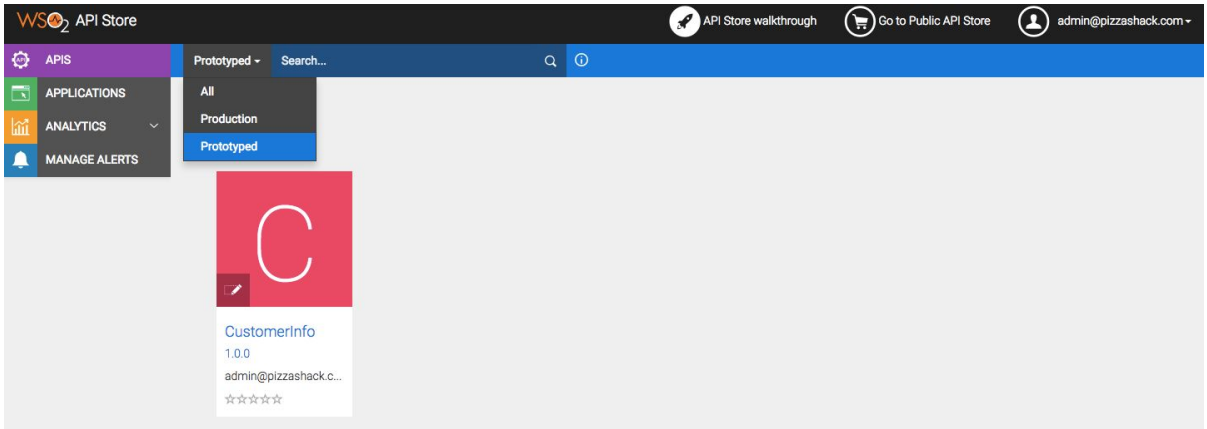


12. Click **Deploy as Prototype**.



13. Click **Go to Overview** and to see the overview of the newly created API.
14. Go to the API Store https://localhost:9443/store/.
15. Go to the **carbon.super** store and note that the newly created API does not appear.
16. Try logging in as admin and note that the API is still not visible.

17. Log in to the **pizzashack.com** store as admin@pizzashack.com and view API under Prototyped APIs.



Expected Outcome

A tenant is created to contain the employees of PizzaShack. An API is created to capture statistical data of customers and this is shared with only this tenant.

Lab: Subscribing to APIs

Training Objective

Learn how to subscribe to the APIs using the store.

Business Scenario

After PizzaShack successfully publishes the APIs other partners who would like to use the PizzaShackAPIs as a base can open the API store and check its contents and subscribe to the API if interested.

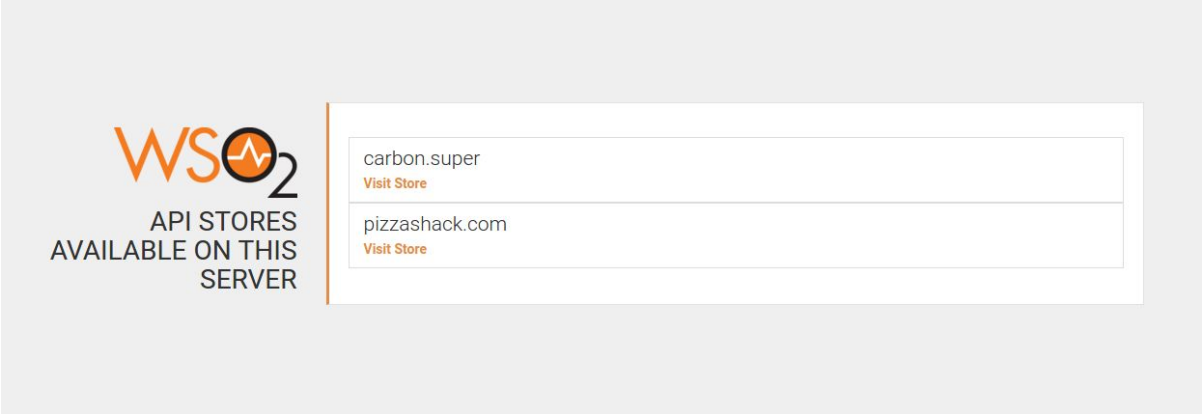
High Level Steps

- Browse the store
- Define users via self-registration
- Subscribe to APIs

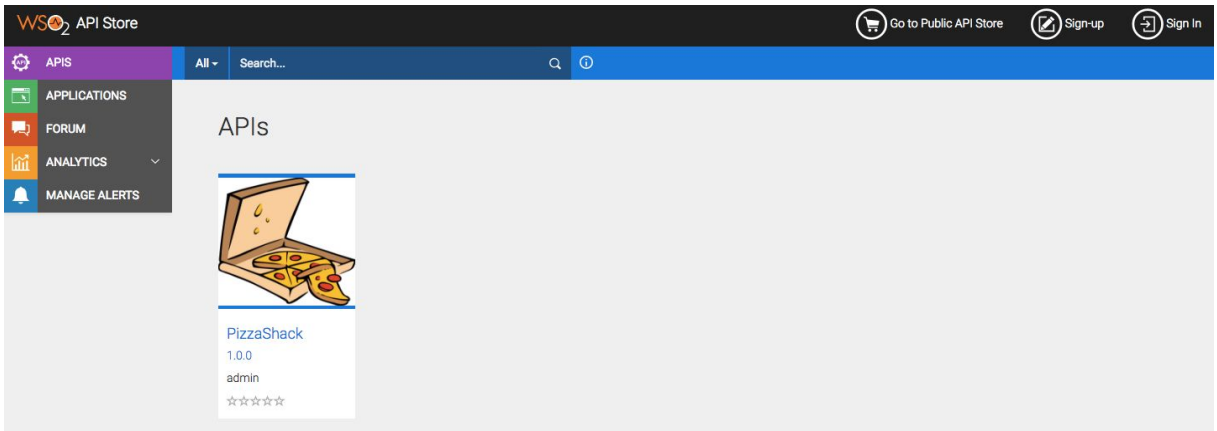
Detailed Instructions

Browse the Store

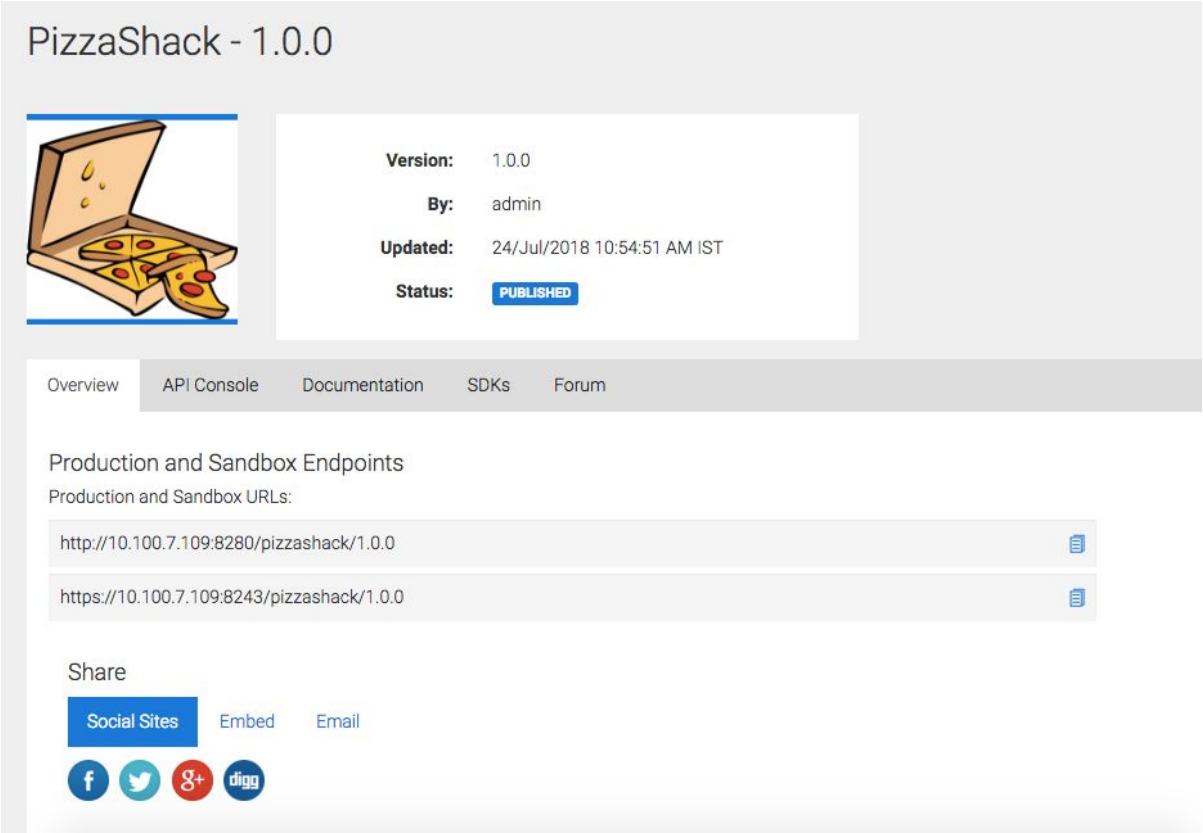
1. To view the API store contents, open the following URL:https://localhost:9443/store.



2. Click carbon.super.
3. Log off if you have already logged in.



4. Click the icon to see the details entered by the API creator:



You can browse the API store and check the documentation without the necessity to provide credentials.

You can search API by their name, context, version or by clicking on the tags to the left. You can also test the API from the API Console, but prior to that, you need to subscribe to the APIs to obtain a security token.

Define Users via Self-Registration

When a user connects to the API store for the first time, they can self-register.

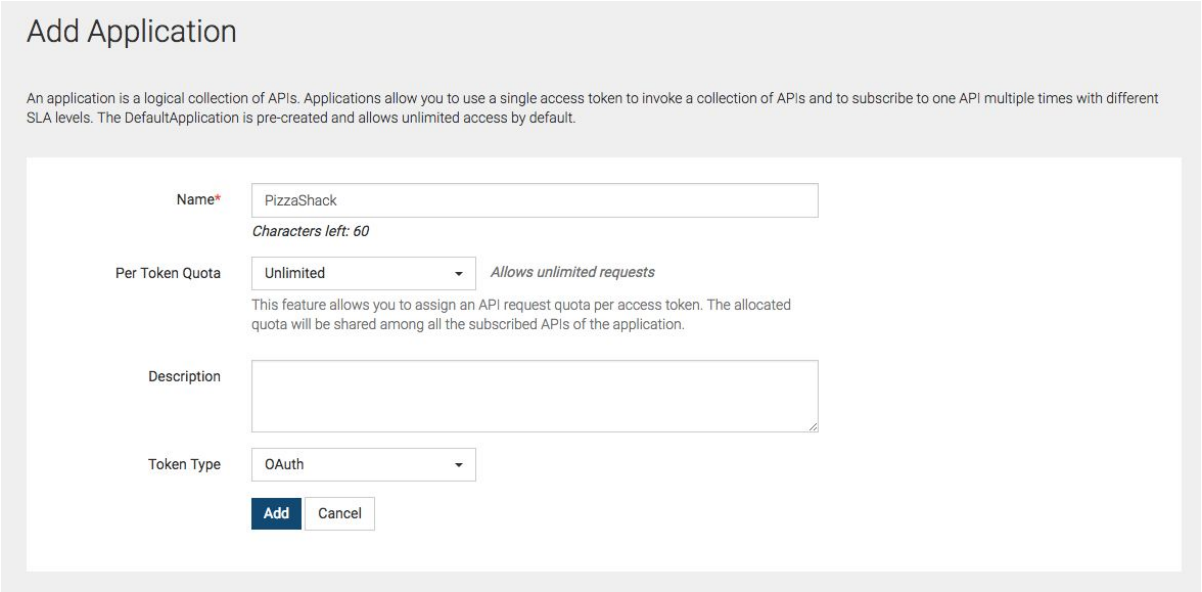
1. While within the carbon.super tenant, click **Sign-Up** at the top right of the window.
2. Fill in the fields as required and click **Submit**.

The **subscriber** role is already defined out of the box, as it is used in the self-registration process.

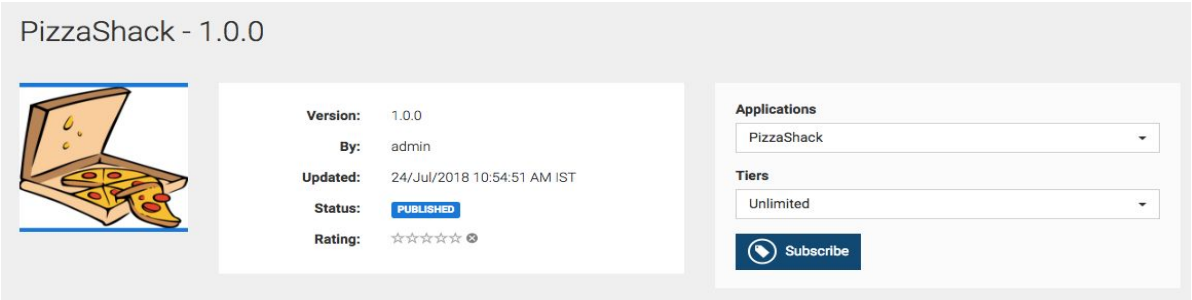
Subscribe to an API

As a consumer, you can subscribe to an API by following those steps:

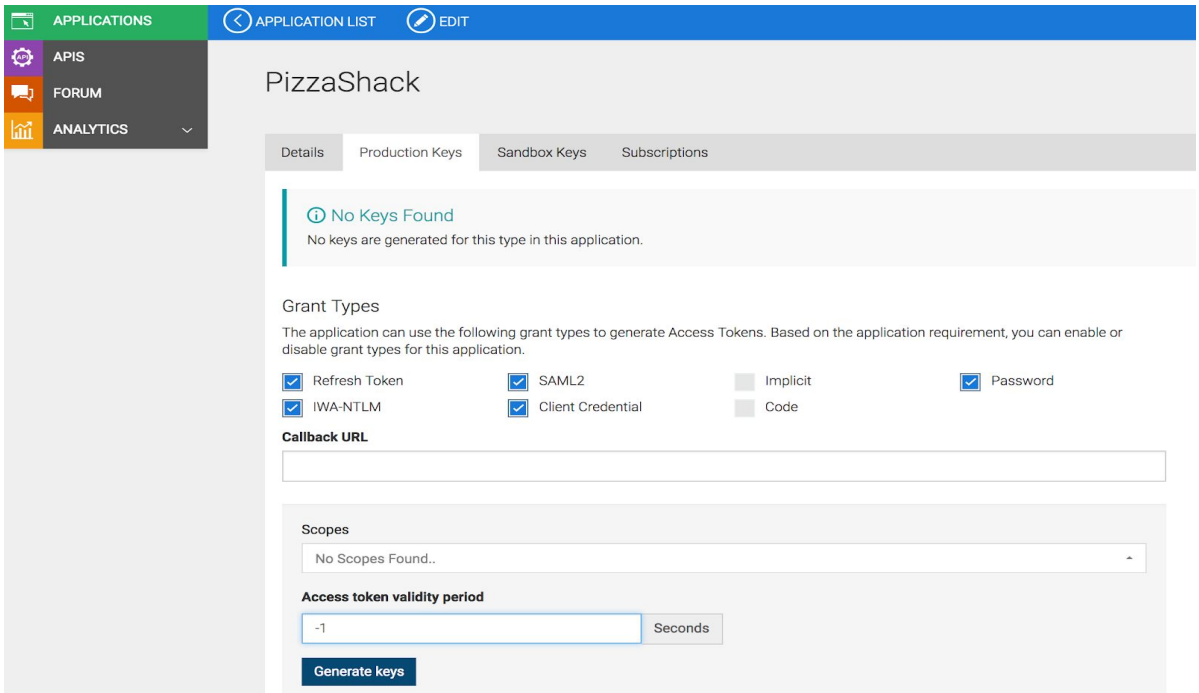
1. Log in to the store using the user created in the above exercise and access the **carbon.super tenant**. You can now see additional information for the API, and set ratings and provide comments.
2. Go to **APPLICATIONS**, click **ADD APPLICATION** and create a PizzaShack application. Select **Unlimited** in the **Per Token Quota** field.



3. Click the **APIs** tab, select the Pizza API, subscribe to the API selecting the PizzaShack application - Select the **Unlimited** tiers level (we need to do several calls in a limited time from the Pizza web application).



4. Click **Subscribe.**
5. Switch to the **APPLICATIONS** page. Select PizzaShack application from the list.
6. Click the **Production Keys** tab.
7. Click **Generate Keys**. (Enter -1 as the **Access token validity period** to make sure that the validity period of the user access token will be unlimited).
8. You have now successfully subscribed to the API and can start using it.



Note: User access tokens have a fixed expiration time, which is set to 60 minutes by default. Before deploying the API-M the default expiration time can be extended by editing the <AccessTokenDefaultValidityPeriod> tag in <PRODUCT_HOME>/repository/conf/identity/identity.xml.

Expected Outcome

As a result of this exercise, a user and application have been created for subscription, the API has been subscribed to, and access tokens have been generated.

Lab: Invoking the API

Training Objective

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

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 OAuth2 access tokens on demand.

High Level Steps

- Test the API
- Deploy and test the PizzaShack Application

Detailed Instructions

Test the API

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

Using cURL this is very simple - 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 copy the **Access Token** in the -**APPLICATIONS** page and add it as the Authorization Bearer.

```
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.

1. Download the pizzashack.war file from the link and copy it to <APIM_HOME>/repository/deployment/server/webapps so that it is deployed.
2. 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:

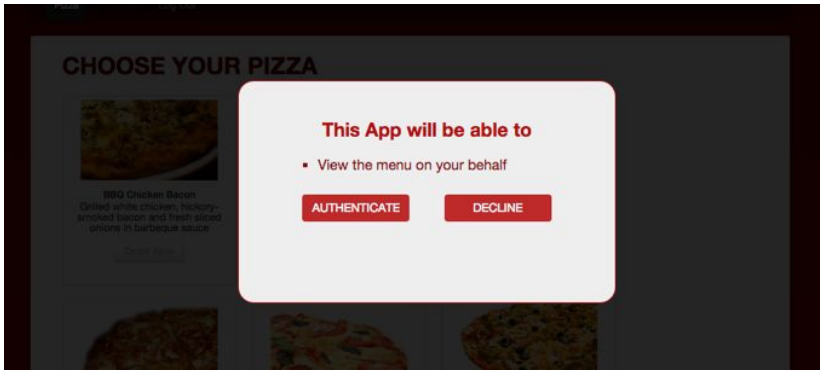
```
</context-param>
<context-param>
        <param-name>consumerKey</param-name>
        <param-value>XXXXXXXXXXXXXXXXXXXXXXXXXXXX</param-value>
```



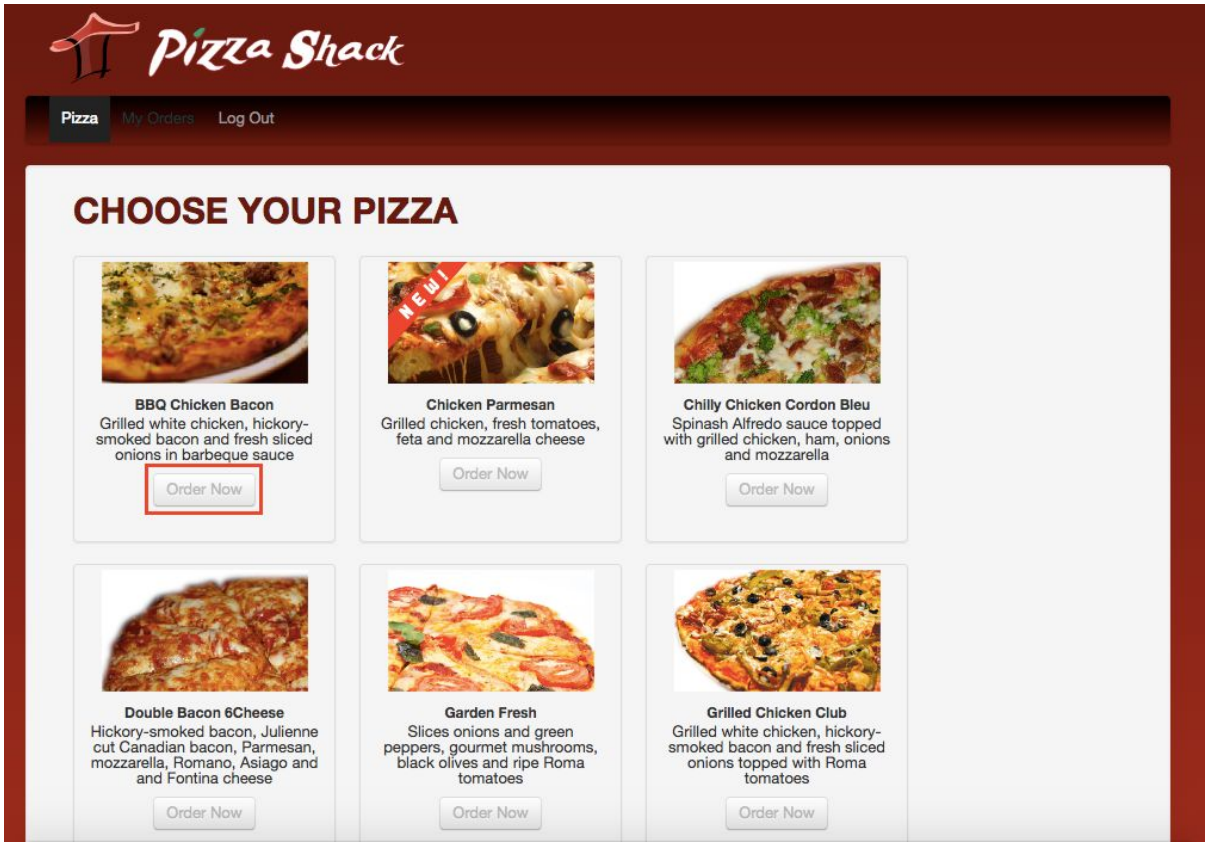
```
</context-param>
<context-param>
        <param-name>consumerSecret</param-name>
        <param-value>YYYYYYYYYYYYYYYYYYYYYYYYYY</param-value>
 </context-param>
```

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

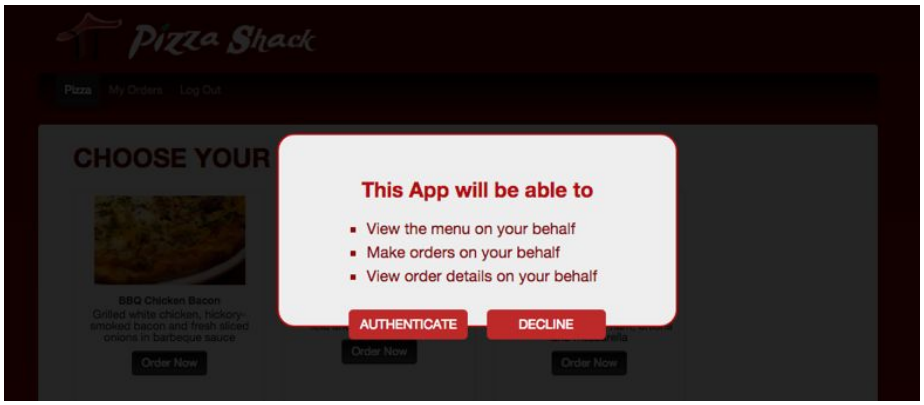
When mike logs in, he will not be able to get a token having the **order_pizza** scope since he doesn't have the **webuser** role. As a result, you will see the screen below.



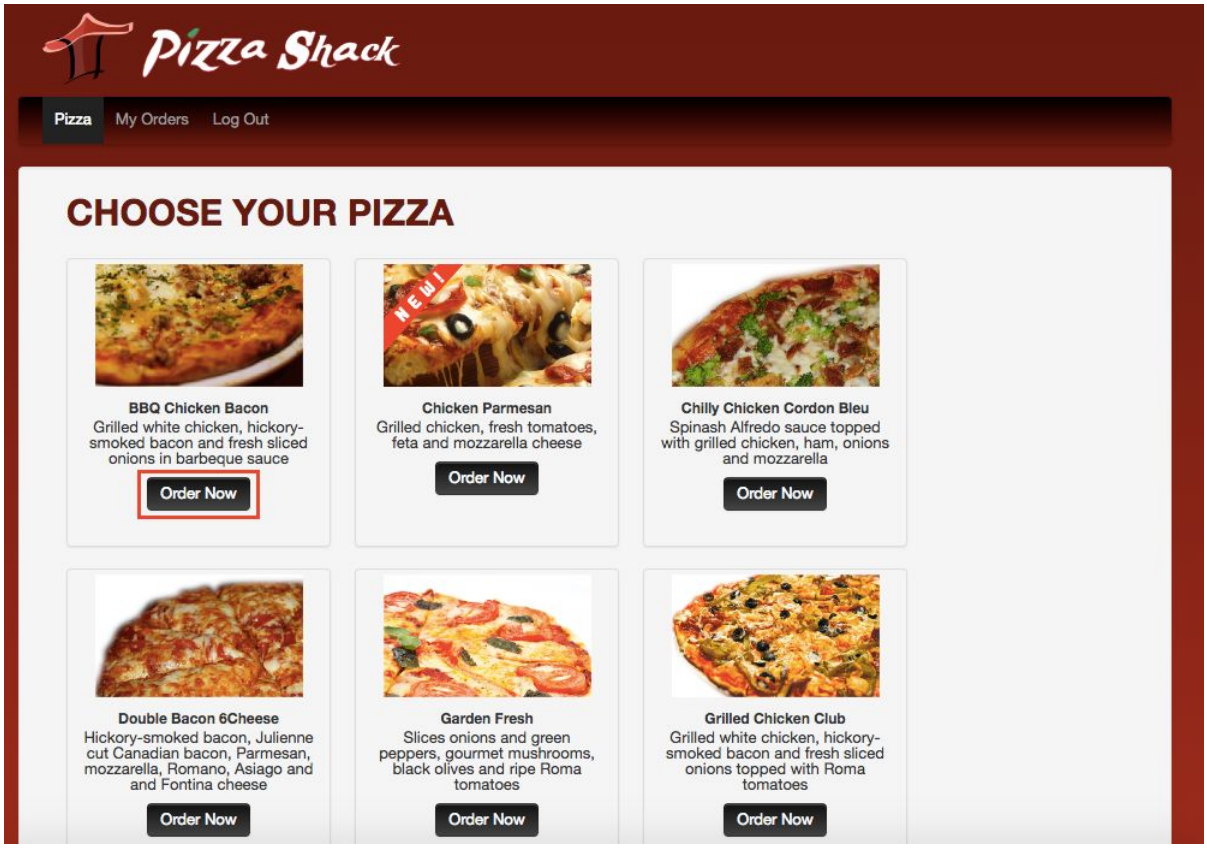
Note that the **Order Now** button is disabled.



6. Log in as john. Since user john has the **webuser** 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 john, you will see the screen below.



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



Expected Outcome

In this exercise, the API was tested using 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.

Lab: Working with Throttling Policies

Training Objective

Add new throttling tiers and define extra properties to throttling tiers using the Admin Portal. Throttling allows you to limit the number of hits to an API during a given period of time.

Business Scenario

PizzaShack's popularity is overwhelming and the amount of requests is increasing so they have decided to allow up to 100 requests per minute. Other than that in a recent analysis they could find out that PizzaShackAPI is getting misused through an application called "Pizzaman" and they want to block all calls from that application.

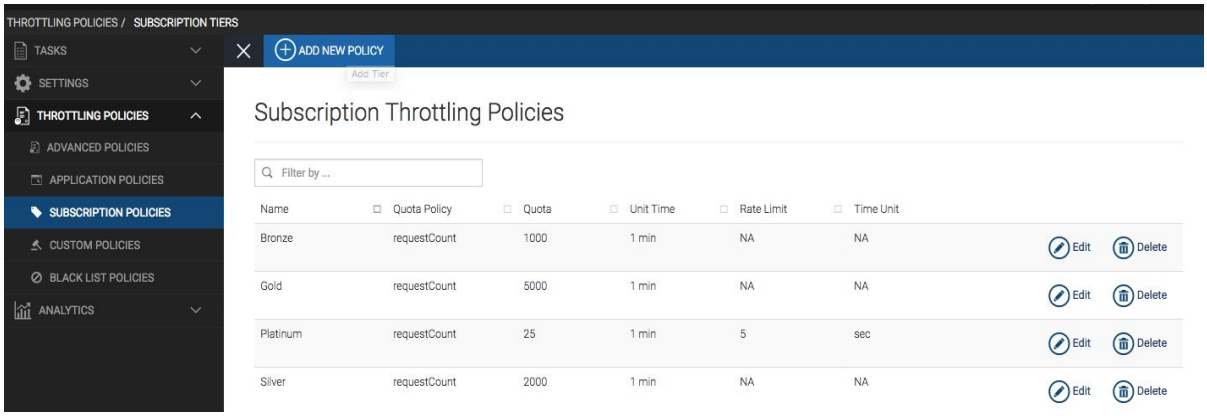
High Level Steps

- Add throttling policy
- Add conditions to advanced throttling
- Block all calls from an application through blacklisting

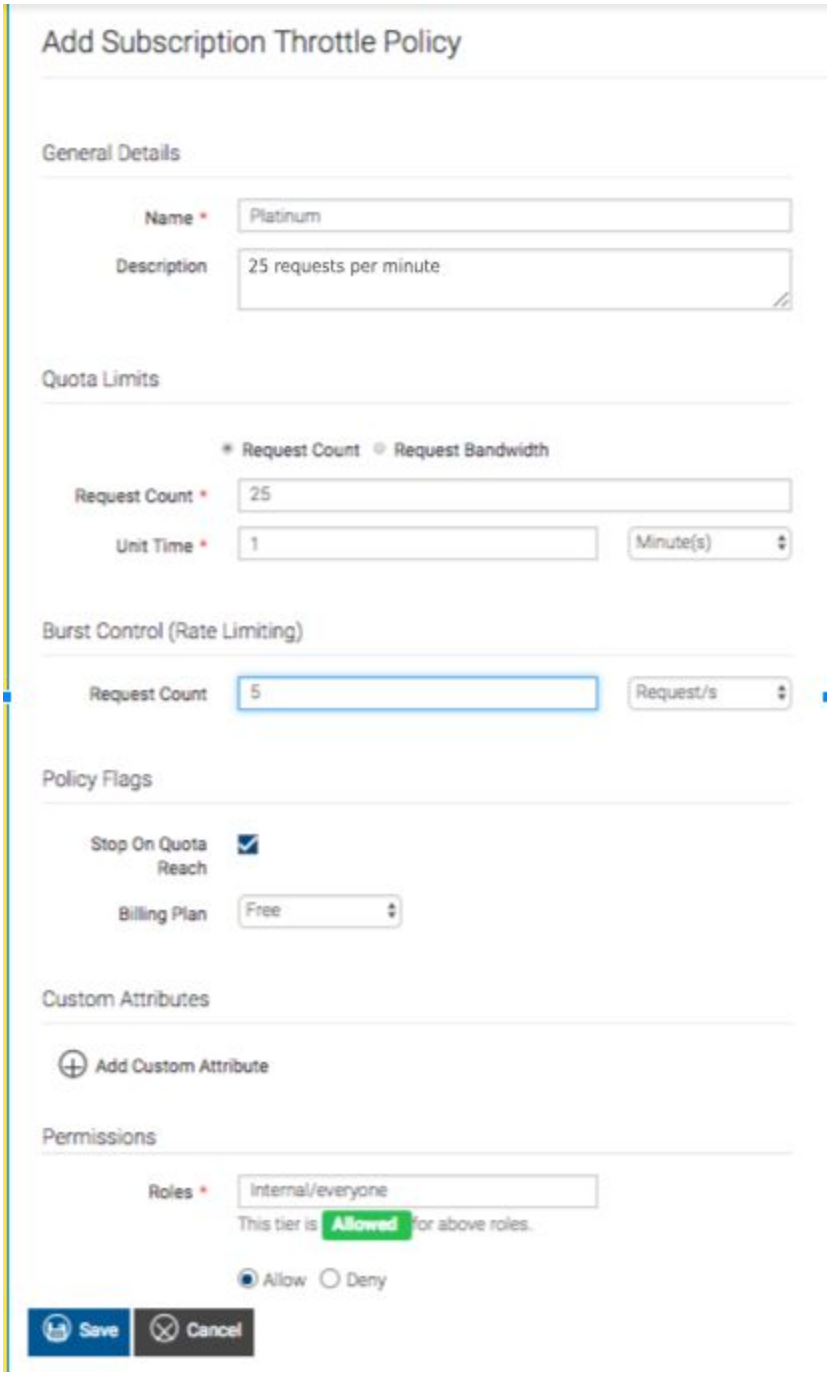
Detailed Instructions

Add Throttling Policy

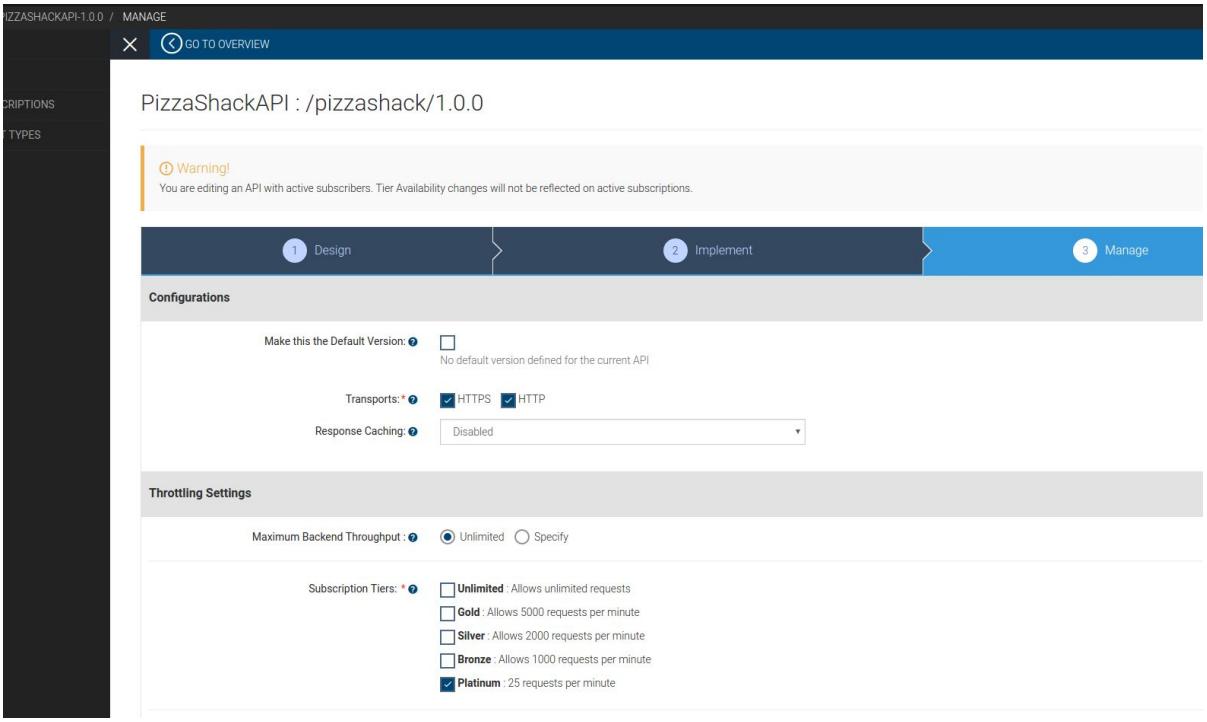
1. Log in to the API Manager Admin Portal (https://localhost:9443/admin/) (admin/admin) and click **THROTTLING POLICIES**.
2. Select **SUBSCRIPTION POLICIES** and **ADD NEW POLICY** at the top.



3. Enter the following information
4.

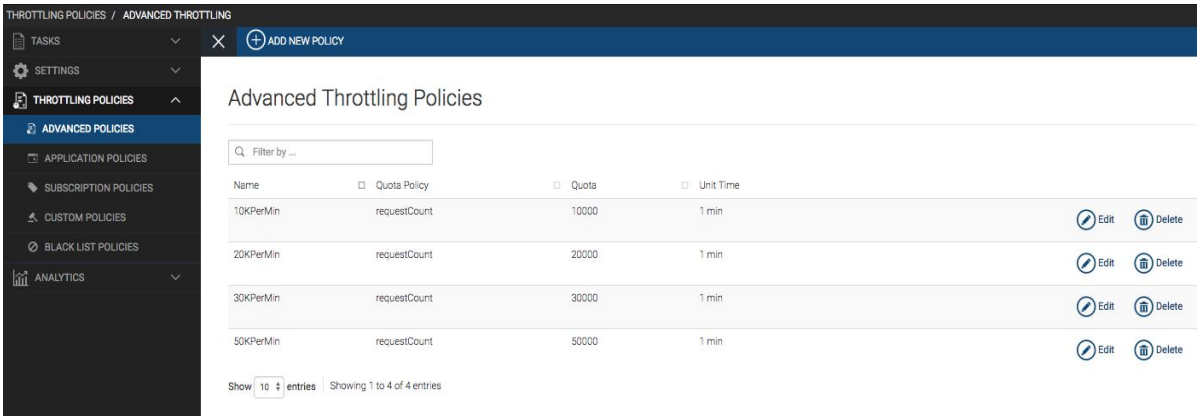


In the API **Publisher**, edit the PizzaShackAPI and note that Platinum can now be selected, **tick the check box of Platinum tier** under **Subscription Tiers** in the **THROTTLING SETTINGS** section, edit and save it, which then will be visible in the API Store whenever a person wants to subscribe to the PizzaShackAPI



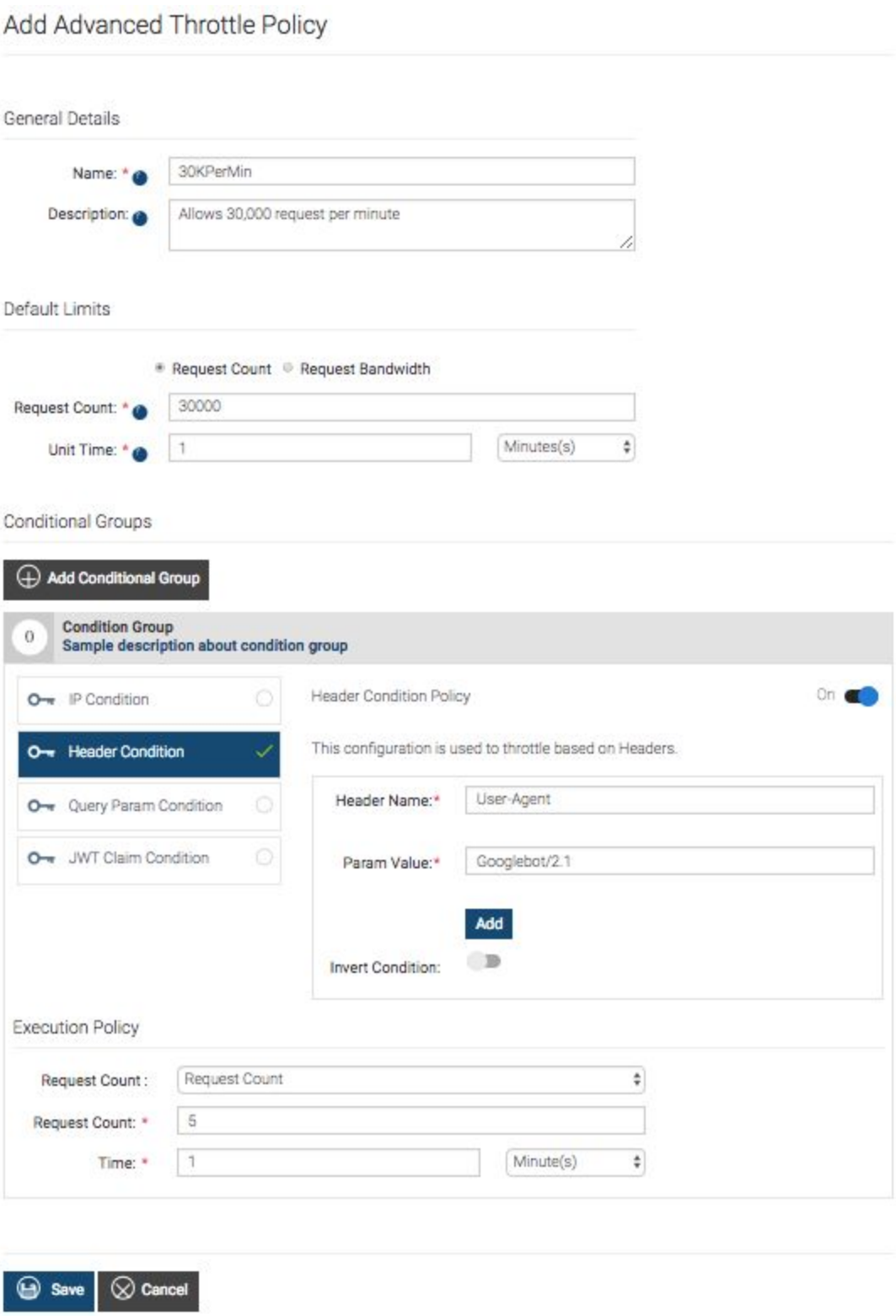
Add Conditions to Advanced Throttling

1. Click **ADVANCED THROTTLING** and select **ADD NEW POLICY** at the top



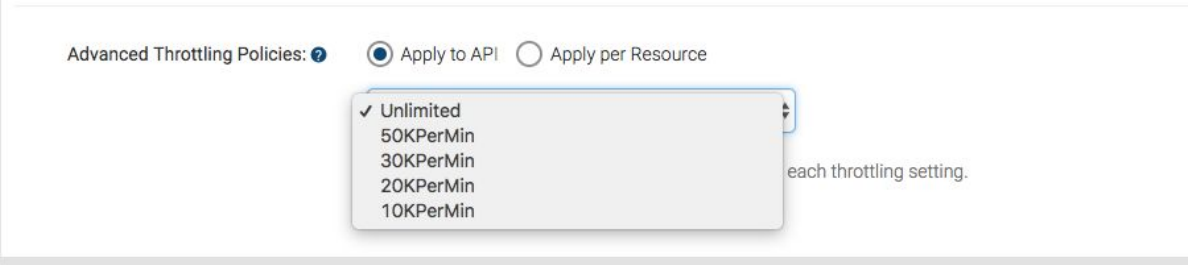
2. Enter the following information and click **Add Conditional Group.** Select **Header Condition** and add the details as show in the image.

   The conditional group is created to apply throttling to users who send the User-Agent header with the value 'Googlebot/2.1', and limits the number of API invocations to 5 requests per minute.



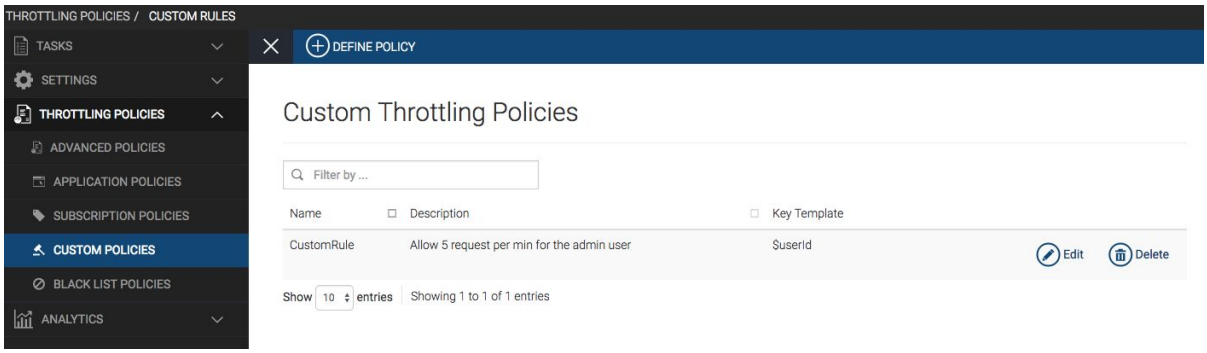
3. Click **Add** to add the Header condition Policy and Click **Save**.

The new advanced throttling policy will be available under the **Advanced Throttling Policies** for API in the **Throttling Settings** section.

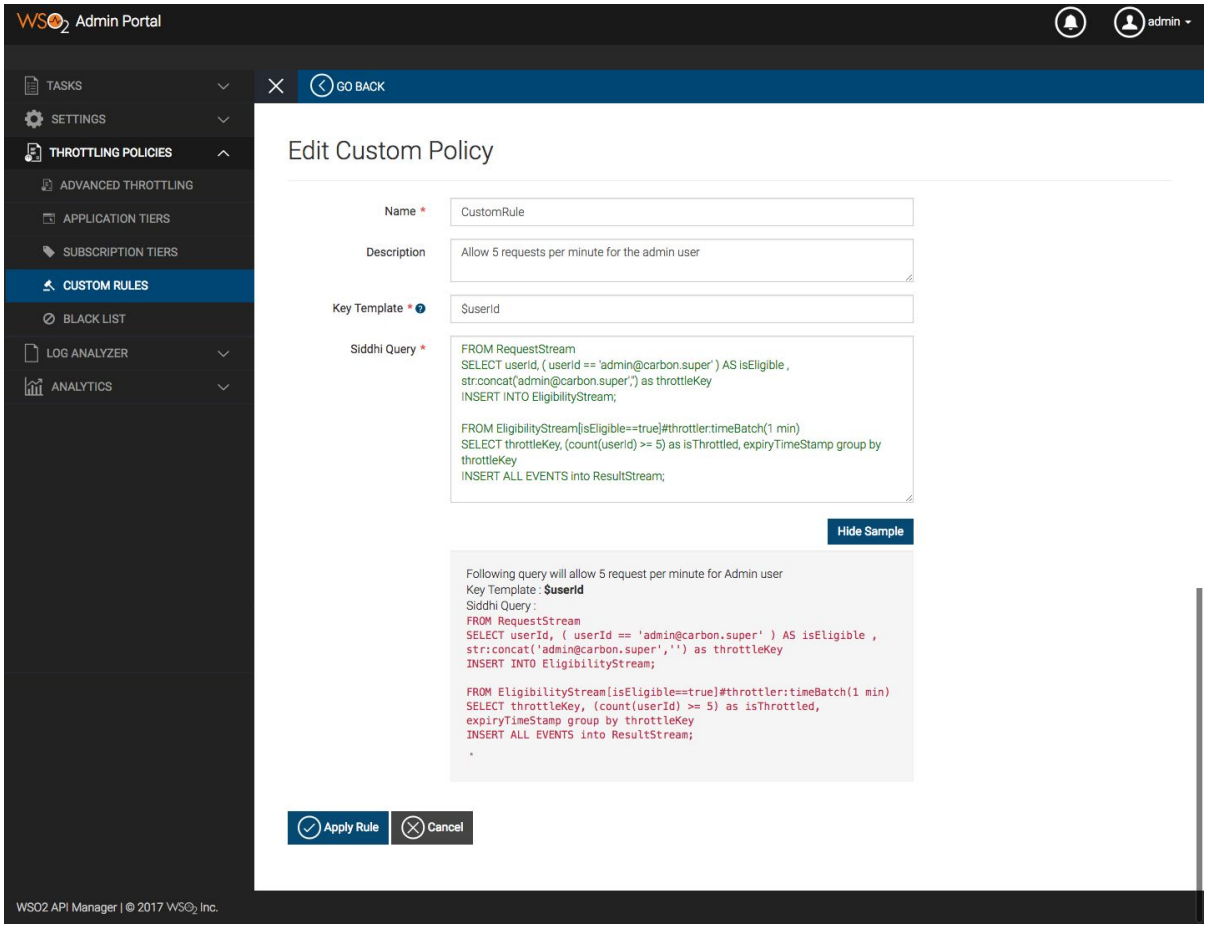


Add Custom Rules

1. Click **CUSTOM POLICIES** and select **DEFINE POLICY**.



2. Enter the following information



Add following details.

Name	CustomPolicy
Description	Allow 5 requests per minute for admin user
Key Template	$userId

3. Paste the query below as the Siddi Query.
(You can get this sample siddi query by clicking on show sample option as well.)

Sample Siddi Query :

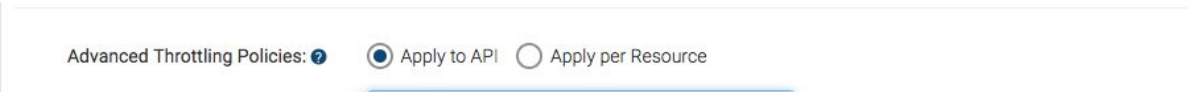
```
FROM RequestStream
SELECT userId, ( userId == 'admin@carbon.super' ) AS isEligible ,
str:concat('admin@carbon.super','') as throttleKey
INSERT INTO EligibilityStream;

FROM EligibilityStream[isEligible==true]#throttler:timeBatch(1
min)
SELECT throttleKey, (count(userId) >= 5) as isThrottled,
expiryTimeStamp group by throttleKey
INSERT ALL EVENTS into ResultStream;
```

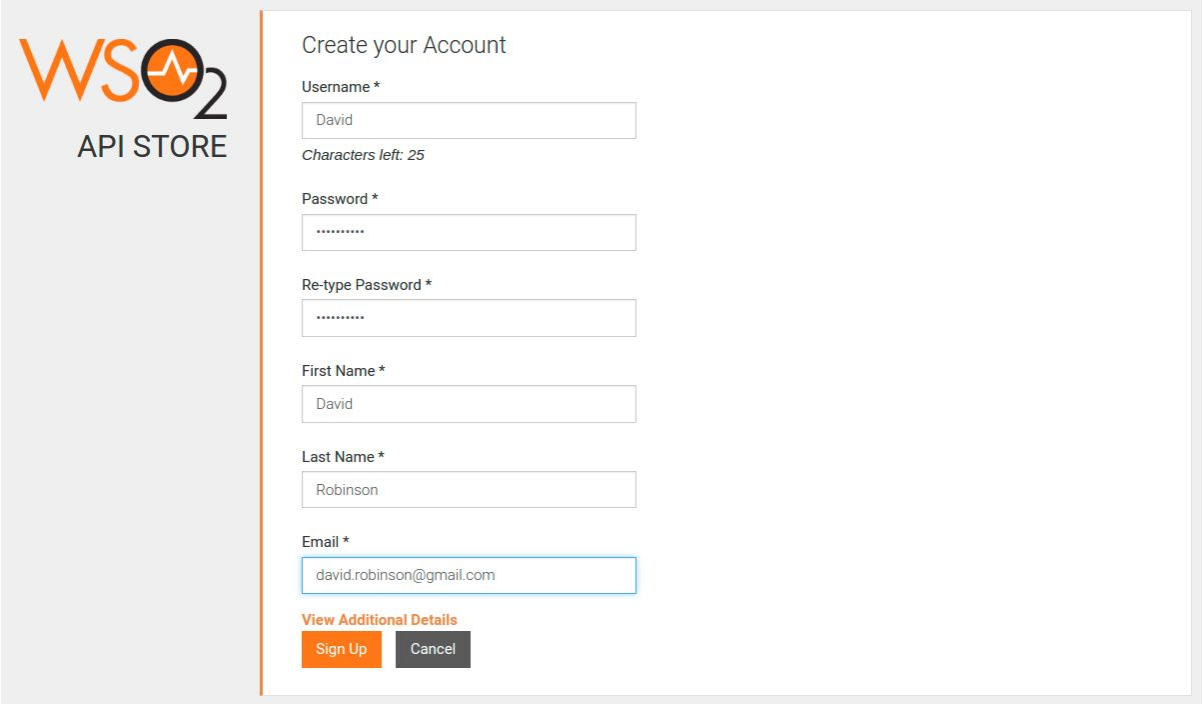
4. Click **Apply Rule**.

Blacklisting Application

1. Log in to API Publisher and go to the edit view of **PizzaShackAPI**.
2. Click **Manage** and make sure the **Apply per Resource** is selected under the **Advanced Throttling Policies**.



3. Click **Save & Publish** if you have made changes to the API.
4. Go to API Store (https://localhost:9443/store) and click **Sign Up**.
5. Self Signup a user with adding details as follows. (Alternatively you can use a user which you have created before using Sign Up option).



6. Login to the APIStore with the last created user's credentials.
7. Select carbon.super tenant
8. Go to **APPLICATIONS** and click **ADD APPLICATION** on the top.
9. Create an Application named "**Pizzaman**".
10. Subscribe to the **"PizzaShackAPI"** through "**Pizzaman**" with "**Platinum"** tier.
11. Generate keys for the application under the Production keys tab.
12. Go to **API Console** of **PizzaShackAPI.** Send a GET request to the Resource "**menu**" using the tryout tool. *

***Note:** You might have to accept the certificate if the above function does not work.

Click on "Not Secure" on the address bar -> select "Certificate" -> drag and drop the certificate image on the popup to your desktop -> open "localhost.cer" in your desktop -> go to certificates on the category bar -> open localhost -> expand "Trust" tab -> Select "when using this certificate" and set it to "Always Trust".

Now repeat step 12 and it should now display the response.

Execute | Clear

Responses

Response content type application/json

Curl

```
curl -k -X GET "https://10.100.7.109:8243/pizzashack/1.0.0/menu" -H "accept: application/json" -H "Authorization: Bearer bb79f466-35f6-3066-be29-572828301bf9"
```

Request URL

```
https://10.100.7.109:8243/pizzashack/1.0.0/menu
```

Server response

Code	Details
200	Response body


```
[
  {
    "name": "BBQ Chicken Bacon",
    "description": "Grilled white chicken, hickory-smoked bacon and fresh sliced onions in barbeque sauce",
    "price": "13.99",
    "icon": "/images/6.png"
  },
  {
    "name": "Chicken Parmesan",
    "description": "Grilled chicken, fresh tomatoes, feta and mozzarella cheese",
    "price": "10.99",
    "icon": "/images/1.png"
  },
  {
    "name": "Chilly Chicken Cordon Bleu",
    "description": "Spinash Alfredo sauce topped with grilled chicken, ham, onions and mozzarella",
    "price": "22.99",
    "icon": "/images/10.png"
  },
  {
    "name": "Double Bacon 6Cheese",
    "description": "Hickory-smoked bacon, Julienne cut Canadian bacon, Parmesan, mozzarella, Romano, Asiago and and Fontina cheese",
    "price": "18.99",
    "icon": "/images/9.png"
  },
  {
    "name": "Garden Fresh",
    "description": "Slices onions and green peppers, gourmet mushrooms, black olives and ripe Roma tomatoes",
    "price": "10.99",
    "icon": "/images/3.png"
  },
  {
    "name": "Grilled Chicken Club",
    "description": "Grilled white chicken, hickory-smoked bacon and fresh sliced onions topped with Roma tomatoes",
    "price": "24.99",
    "icon": "/images/8.png"
  },
  {
    "name": "Hawaiian BBQ Chicken",
    "description": "Grilled white chicken, hickory-smoked bacon, barbeque sauce topped with sweet pine-apple",
    "price": "22.99",
    "icon": "/images/7.png"
  },
  {
    "name": "Spicy Italian",
    "description": "Pepperoni and a double portion of spicy Italian sausage",
    "price": "11.99",
    "icon": "/images/2.png"
  },
  {
    "name": "Spinach Alfredo",
    "description": "Rich and creamy blend of spinach and garlic Parmesan with Alfredo sauce",
    "price": "20.99",
    "icon": "/images/5.png"
  },
  {
    "name": "Tuscan Six Cheese",
    "description": "Six cheese blend of mozzarella, Parmesan, Romano, Asiago and Fontina",
    "price": "20.99",
    "icon": "/images/4.png"
  }
]
```


Response headers

```
content-type: application/json
```

Responses

Alternatively you can use the following CURL/Rest API command

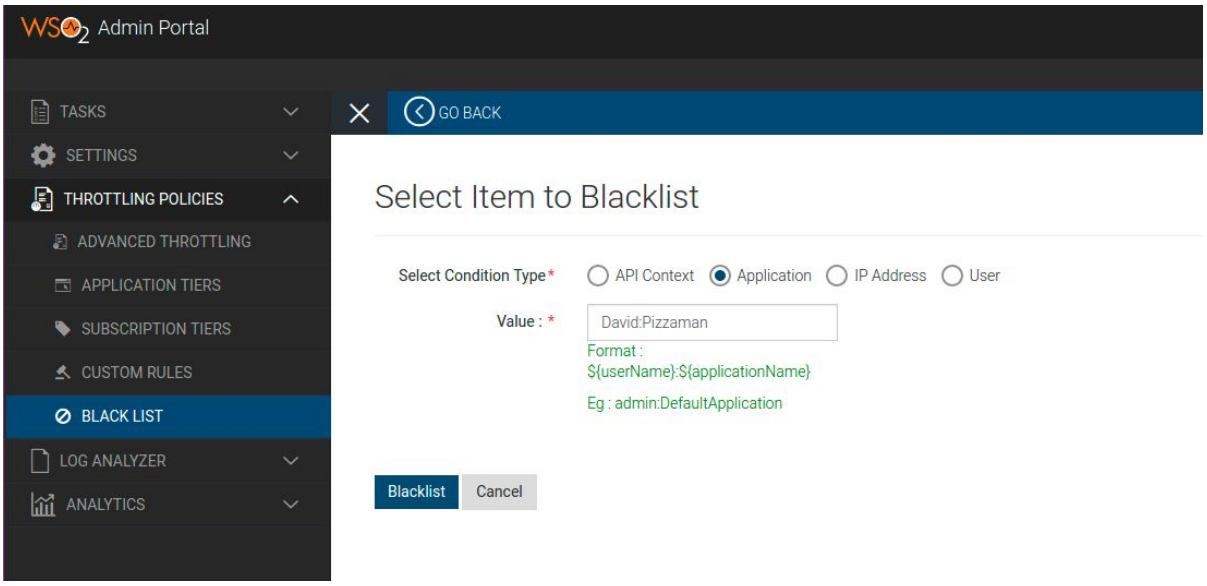
Replace the Authorization Bearer token with the access token retrieved when generating the Keys for **Pizzaman** Application.

```
curl -k -X GET --header 'Accept: application/json' --header 'Authorization:
Bearer 31bdb476-6b28-360e-a61e-25845b4e4921'
'https://localhost:8243/pizzashack/1.0.0/menu'
```

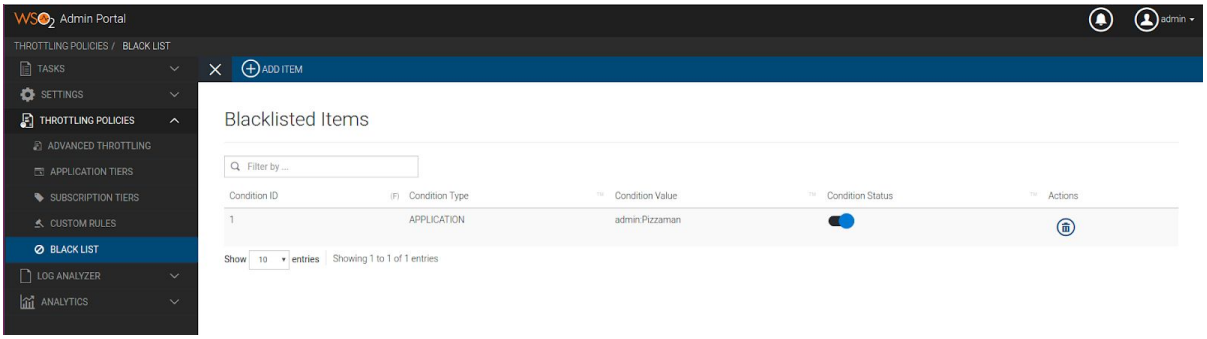
Method : GET
URL : http://localhost:8280/pizzashack/1.0.0/menu
Authorization tab - Type : Bearer Token,
Token : “d919d61a-8ef4-3059-b28e-9f38023aa306”

Application PizzaMan can now successfully invoke the API.

13. Now login to the API Manager Admin Portal (https://localhost:9443/admin/) and click **THROTTLING POLICIES**.
14. Click on **BLACK LIST** and add the application name with the username which need to blacklist in following format.
    `<username>:<applicationName>`



Blacklisted Application will be listed as follows.



15. Now go to API Store again and invoke the **PizzaShackAPI** same as in step 11 and 12 using the **Pizzaman** application.

You should receive the following response.

```
{
  "fault": {
    "code": 900805,
    "message": "Message blocked",
    "description": "You have been blocked from accessing the
resource"
  }
}
```

Expected Outcome

Your new subscription tier (Platinum) is now successfully saved as an execution plan used by WSO2 API Manager. You can view this new throttle tier available for selection when creating a new API through the API Publisher or when editing an existing API.

Your new advanced throttling policy 30KPerMin, with conditional throttling groups, is now successfully saved as a throttling policy. You can apply to the whole API or selected resources.

Invoking the PizzaShackAPI by the specific user (David) through Pizzaman API is now blocked as the application Pizzaman is blacklisted.

Lab: Analyze Runtime Statistics

Training Objective

Set up WSO2 API Manager Analytics server to collect and analyze runtime statistics from the API Manager.

Business Scenario

PizzaShack Limited needs to monitor the use of their online portal and want to generate statistics about how many times consumers access the API.

High Level Steps

- Configure WSO2 API Manager
- View published statistics

Detailed Instructions

Configure WSO2 API Manager

1. Open the <API-M_HOME>/repository/conf/api-manager.xml file and set the <Enabled> element in the <Analytics> section to true. Shut down the API-M server. API Manager Analytics comes with a default port offset of 1. It points to an H2 RDBMS database which is used by the API Manager.

2. To run the setup, extract API Manager Analytics 2.6.0.
3. Start the WSO2 APIM Analytics server, and then start the API Manager server. For more information, see link or download it here

View Published Statistics

1. Invoke the API.
2. Log in to the API Publisher.
3. Click **Analytics** and click each link to view the statistics.

Expected Outcome

As a result of this exercise, events are generated based on the API invocations and stored in the RDBMS tables shared with the API Manager and API Manager Analytics server.

Lab: Managing Alerts with Real-Time Analytics

Training Objective

Generate alerts for a scenario when the tier limit is hit frequently.

Business Scenario

PizzaShack Limited needs to generate alerts if users exceed their tier limits frequently.

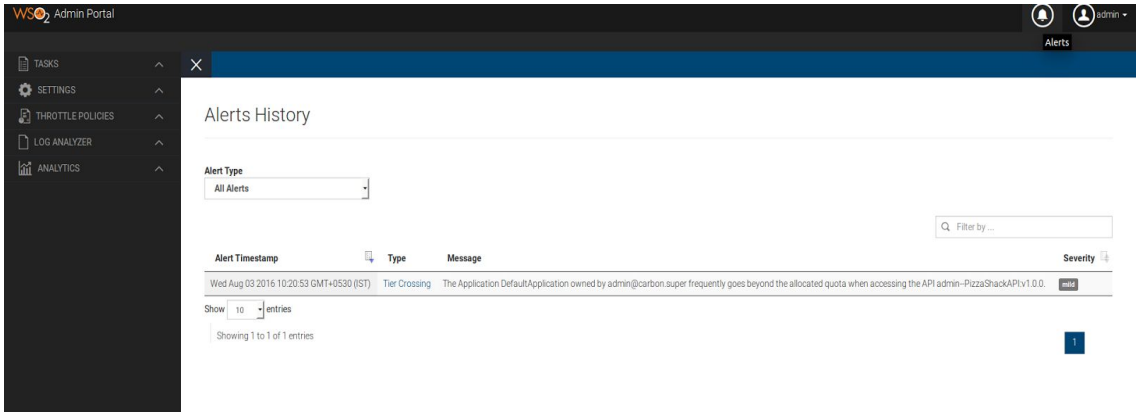
High Level Steps

- Generate and view alerts
- Configure alert generation related parameters

Generate Alerts

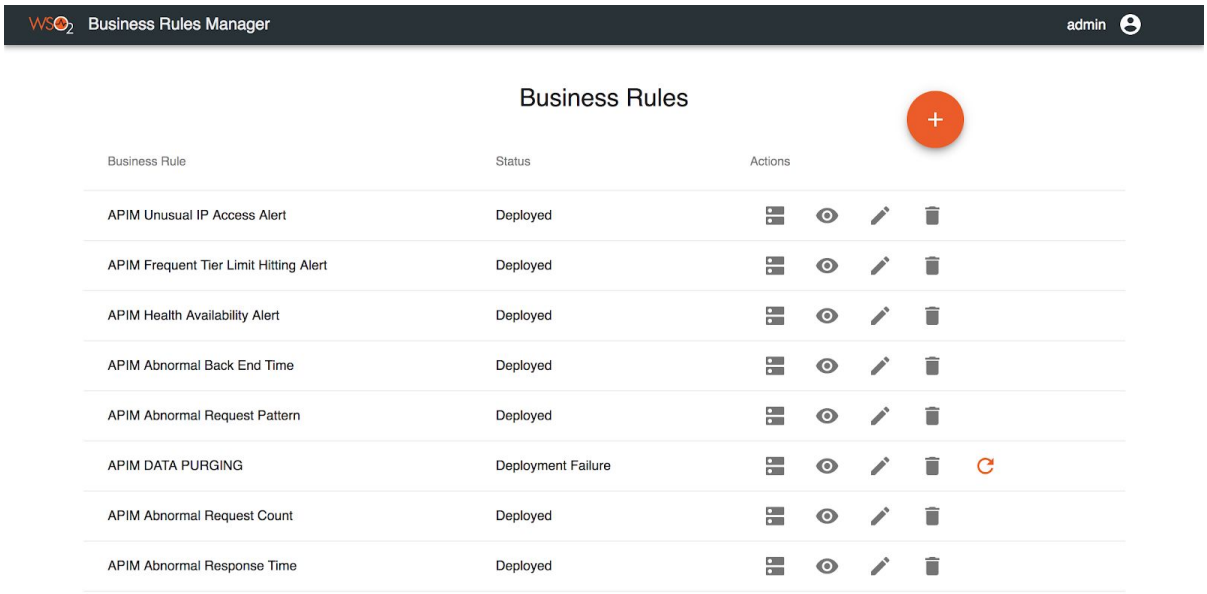
Note: API Manager and API Manager Analytics must be configured for analytics. This has been covered in the previous exercise.

1. Create a subscriber level throttling tier with a small number of requests per minute. E.g., 2 requests per minute.
2. Log in to the API Publisher.
3. Click on the PizzaShackAPI and click **EDIT API**.
4. Click **Manage**.
5. Apply the new throttling tier and click **Save and Publish**.
6. Log in to API Store and select the PizzaShackAPI.
7. Subscribe to an application using the new tier.
8. Invoke the API rapidly till it throttles out. After 2 requests, you should get a throttled out message. Keep on doing request (more than 10) to generate an alert (by default an alert is generated when there are 10 alerts more than the limit).
9. Login to Admin portal (https://localhost:9443/admin/) and select the alert icon on the top right corner and you will see a generated alert.



Configure Alert Generation Related Parameters

1. Log in to WSO2 API Manager Analytics carbon console (https://localhost:9444/carbon).
2. Go to the Business Rules and Status Dashboard (e.g., https://localhost:9643/business-rules).
3. In the **Business Rules Manager**, select **APIMAnalytics**. This will open a configuration page for all the alert types.



4. To edit parameters related to the frequent tier limit hitting alert click **Edit** in the **FrequentTierLimitHitting** section.

Expected Outcome

As a result of this exercise, Throttled out events are generated based on the API invocations and once the pre-defined tier crossing limit is exceeded, an alert is generated.

Lab: Using Published APIs

Training Objective

In this section you will learn how to invoke, manage and control published APIs through the terminal using cURL.

Business Scenario

PizzaShack Limited needs to improve their delivery time in order to provide a better service. They want to use the Google Directions API to assist the delivery team to identify routes with traffic, find the best possible route and reduce delays in finding the customer's location. They have also decided to improve their PizzaShack web portal in order to call the Google Direction API via API Manager to show the best route information.

High Level Steps

- Create published API
- Publish new API
- Create new application
- Create new subscription

Detailed Instructions

Create Published API

1. Create the payload.json file in the <API-M_HOME>/bin folder with the following text and save.

```
{
    "callbackUrl": "www.google.lk",
    "clientName": "rest_api_publisher",
    "tokenScope": "Production",
    "owner": "admin",
    "grantType": "password refresh_token",
    "saasApp": true
}
```

2. Open a Command Line Interface.
3. Navigate to the {API-M_HOME/bin] folder using the command.
4. Give the cURL command for client registration. (Make sure the API Manager server is running before doing this. Also change 'true' to 'false' in the Analytics section inside the api-manager.xml file, if you are not using the Analytics server anymore)

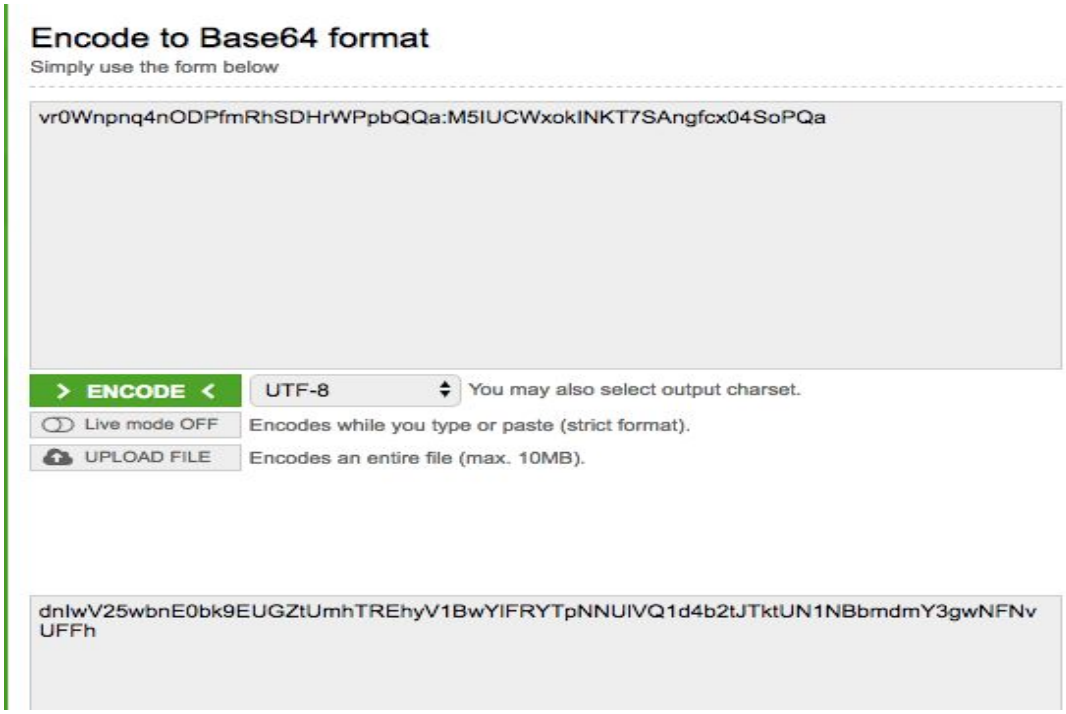
```
curl -k -X POST -H "Authorization: Basic YWRtaW46YWRtaW4="
-H "Content-Type: application/json" -d @payload.json
https://localhost:9443/client-registration/v0.14/register
```

Method : Post
URL : https://localhost:9443/client-registration/v0.14/register
Headers tab - Key 1: Content-Type , Value : application/json
Key 2: Authorization, Value : Basic "64encode admin:admin = YWRtaW46YWRtaW4="
Body tab - paste payload.json code and execute

A response similar to the following is displayed.


```
{"clientId":"3O49fv5wrUw9OE0H_bbK0seUBhwa","clientName":"admi
n_rest_api_publisher","callBackURL":"www.google.lk","clientSe
cret":"5ZahdbfkBfTEVh2rJID2gUPmrcAa","isSaasApplication":true
,"appOwner":null,"jsonString":"{\"grant_types\":\"password
refresh_token\"}","jsonAppAttribute":"{}","tokenType":null}
```


5. Copy the clientId and clientSecret from the console and encode them to generate a key using https://www.base64encode.org/ or any other encoder.



Note : Add a colon (:) between the clientId and clientSecret on Base64.

6. Type the following authorization invocation cURL command on the terminal with the encoded clientId and clientSecret for the Authorization Basic value and scope=apim_create as the scope.

```
curl -k -d
"grant_type=password&username=admin&password=admin&scope
=apim:api_create" -H "Authorization: Basic <encoded
value clientid:clientsecret>"
https://127.0.0.1:8243/token
```

Method : POST
URL : https://127.0.0.1:8243/token
Headers tab - Key 1: Authorization, Value: Basic <encoded value of clientid:clientsecret>
Body tab - Key 1: grant_type, Value: password

Key 2: username, Value: admin, Key 3:password, Value:admin
Key 4: scope, Value: apim:api_create
Authorization tab - Inherit auth from parent

A response similar to the following is displayed.


```
{"access_token":"5951cca5-7dfc-3b48-9104-e5b73c6c4d62","refresh_token":"f029e3d5-58d4-3127-bdbc-cb7c012d663f","scope":"apim:api_create","token_type":"Bearer","expires_in":3600}
```


Note : The scope is given depending on the requirement .A new token is required each time a different scope is used and each token is valid only for 1 hour

7. Create the data.json file in the [API-M_HOME]/bin folder.
8. Add the following json to data.json and save


```
{
   "sequences": [],
   "tiers":    [
      "Bronze",
      "Gold"
   ],
   "visibility": "PUBLIC",
   "visibleRoles": [],
   "visibleTenants": [],
   "cacheTimeout": 300,
   "endpointConfig":
"{\"production_endpoints\":{\"url\":\"http://maps.google.com/maps/api/directions/\",\"config\":null},\"endpoint_type\":\"http\"}",
   "subscriptionAvailability": null,
   "subscriptionAvailableTenants": [],
   "destinationStatsEnabled": "Disabled",
   "apiDefinition":
"{\"paths\":{\"\/*\":{\"get\":{\"x-auth-type\":\"Application\",\"x-throttling-tier\":\"Unlimited\",\"responses\":{\"200\":{\"description\":\"OK\"}}}}},\"x-wso2-security\":{\"apim\":{\"x-wso2-scopes\":[]}},\"swagger\":\"2.0\",\"info\":{\"title\":\"GoogleDirectionsAPI\",\"description\":\"Calculates directions between locations\",\"contact\":{\"email\":\"ApiPublisher@pizzashack.com\",\"name\":\"ApiPublisher\"},\"version\":\"Beta\"}}",
   "responseCaching": "Disabled",
   "isDefaultVersion": true,
   "gatewayEnvironments": "Production and Sandbox",
   "businessInformation":    {
      "technicalOwner": "ApiCreator",
      "technicalOwnerEmail": "ApiCreator@pizzashack.com",
      "businessOwner": "ApiPublisher",
      "businessOwnerEmail": "ApiPublisher@pizzashack.com"
```



```
    },
    "transport":    [
       "http",
       "https"
    ],
    "tags":    [
       "phone",
       "multimedia",
       "mobile"
    ],
    "provider": "admin",
    "version": "Beta",
    "description": "Calculates directions between locations",
    "name": "GoogleDirectionsAPI",
    "context": "/googledirections"
 }
```

9. Run the following cURL command to create the api using data.json. Type the access token obtained as the Authorization Bearer value.

```
curl -k -H "Authorization: Bearer <access token>" -H
"Content-Type: application/json" -X POST -d @data.json
https://127.0.0.1:9443/api/am/publisher/v0.14/apis
```

Method : POST
URL : https://127.0.0.1:9443/api/am/publisher/v0.14/apis
Authorization tab - Type : Inherit auth from parent
Header tab - key 1: Authorization - Bearer <access token>
Key 2: Content-Type - application/json
Body - raw: paste data.json file code

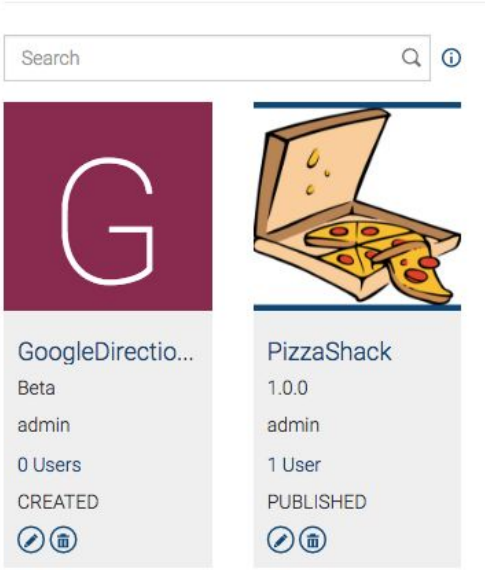
10. The response which includes the id of the API will be displayed.


```
{"id":"f80740db-27ef-4b39-aa92-c8374993e92d","name":"Goog
leDirectionsAPI","description":"Calculates directions
between
locations","context":"/googledirections","version":"Beta"
,"provider":"admin","status":"CREATED","thumbnailUri":nul
l,"apiDefinition":"{\"paths\":{\"/*\":{\"get\":{\"x-auth-
type\":\"Application\",\"x-throttling-tier\":\"Unlimited\
",\"responses\":{\"200\":{\"description\":\"OK\"}}}}},\"x
-wso2-security\":{\"apim\":{\"x-wso2-scopes\":[]}},\"swag
ger\":\"2.0\",\"info\":{\"title\":\"GoogleDirectionsAPI\"
```



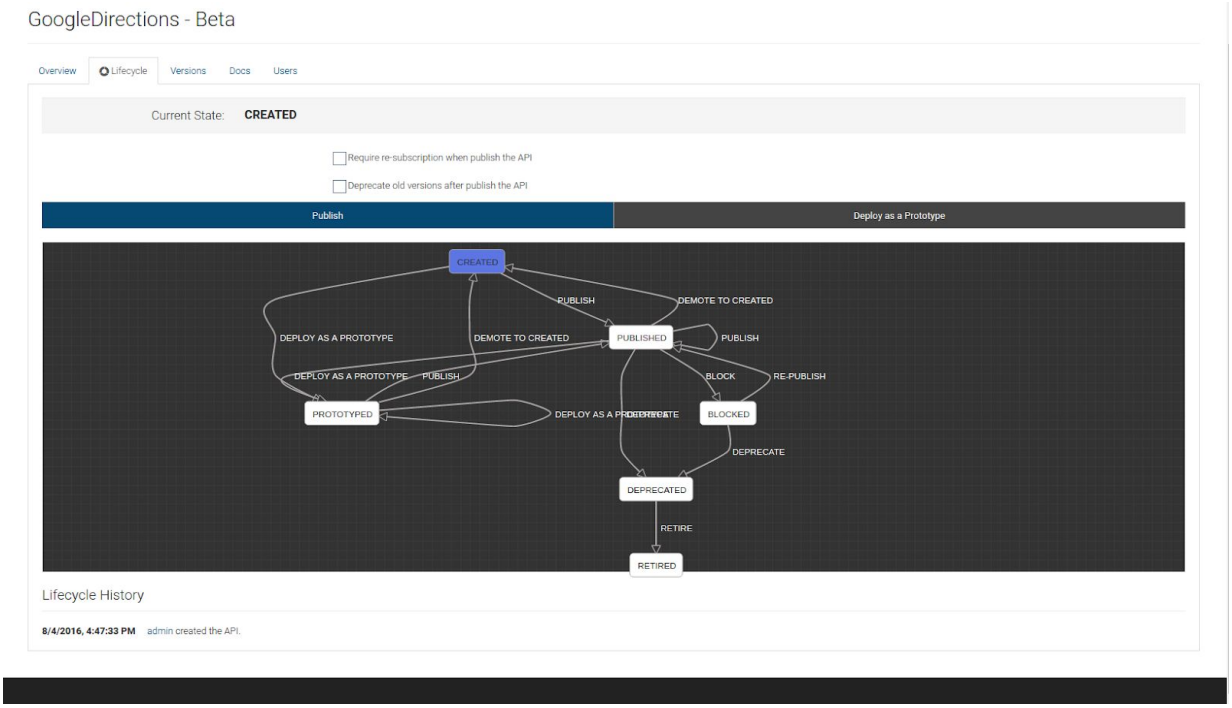
```
,\"description\":\"Calculates directions between
locations\",\"contact\":{\"email\":\"ApiPublisher@pizzash
ack.com\",\"name\":\"ApiPublisher\"},\"version\":\"Beta\"
}}","wsdlUri":null,"responseCaching":"Disabled","cacheTim
eout":300,"destinationStatsEnabled":null,"isDefaultVersio
n":true,"type":"HTTP","transport":["http","https"],"tags"
:["multimedia","phone","mobile"],"tiers":["Bronze","Gold"
],"apiLevelPolicy":null,"authorizationHeader":null,"maxTp
s":null,"visibility":"PUBLIC","visibleRoles":[],"visibleT
enants":[],"endpointConfig":"{\"production_endpoints\":{\
"url\":\"http://maps.google.com/maps/api/directions/\",\"
config\":null},\"endpoint_type\":\"http\"}","endpointSecu
rity":null,"gatewayEnvironments":"Production and
Sandbox","labels":[],"sequences":[],"subscriptionAvailabi
lity":null,"subscriptionAvailableTenants":[],"additionalP
roperties":{},"accessControl":"NONE","accessControlRoles"
:[],"businessInformation":{"businessOwner":"ApiPublisher"
,"businessOwnerEmail":"ApiPublisher@pizzashack.com","tech
nicalOwner":"ApiCreator","technicalOwnerEmail":"ApiCreato
r@pizzashack.com"},"corsConfiguration":{"corsConfiguratio
nEnabled":false,"accessControlAllowOrigins":["*"],"access
ControlAllowCredentials":false,"accessControlAllowHeaders
":["authorization","Access-Control-Allow-Origin","Content
-Type","SOAPAction"],"accessControlAllowMethods":["GET","
PUT","POST","DELETE","PATCH","OPTIONS"]}}
```


11. Refresh https://localhost:9443/publisher/ and the GoogleDirectionAPI will be displayed on the dashboard.



Publish API

1. Log in as admin to https://localhost:9443/publisher.
2. Click on GoogleDirectionsAPI and select the **Lifecycle** tab. The lifecycle will be indicated as **Created**.



3. Go to the terminal and invoke a new authorization token using the following cURL command.with the previously encoded string as the Authorization Basic value and scope=apim_publish.

```
curl -k -d
"grant_type=password&username=admin&password=admin&scope=apim:api_publ
ish" -H "Authorization: Basic <encoded value of
clientid:clientsecret>" https://127.0.0.1:8243/token
```

Method : POST
URL : https://127.0.0.1:9443/api/am/publisher/v0.14/apis
Headers tab - Key 1: Authorization, Value: Basic <encoded value of clientid:clientsecret>
Body tab - Key 1: grant_type, Value: password
Key 2: username, Value: admin, Key 3:password, Value:admin
Key 4: scope, Value: apim:api_publish
Authorization tab - Inherit auth from parent

A response similar to the following is displayed.

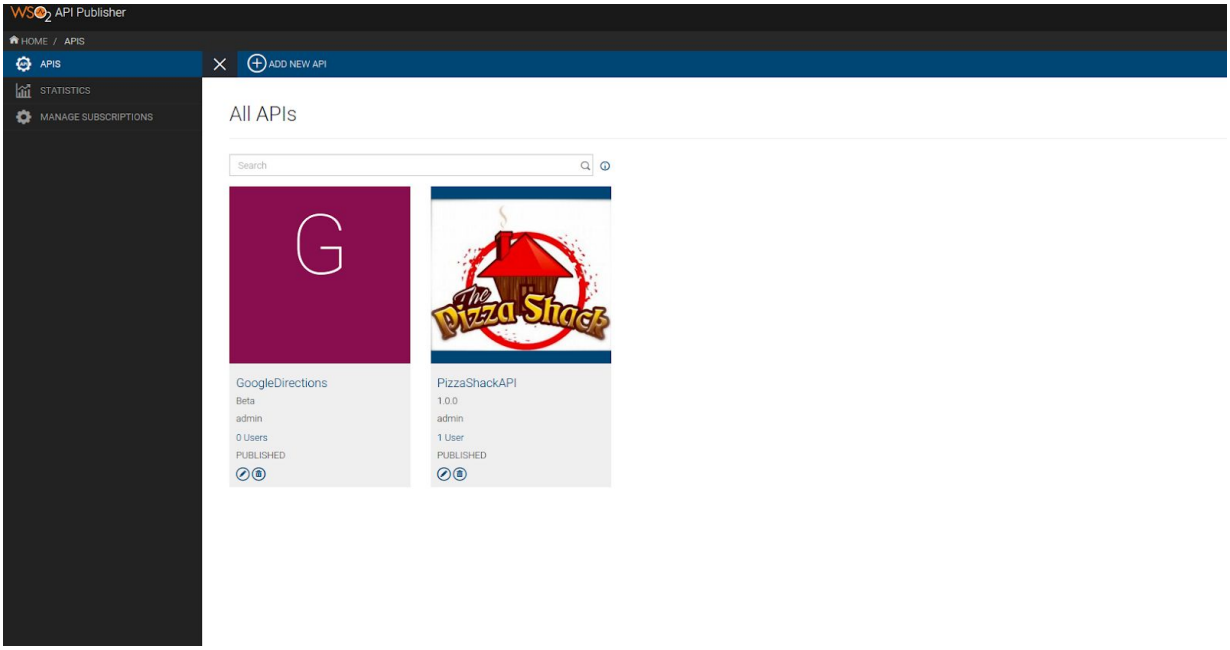
```
{"access_token":"d90dcb86-3bec-323f-98a2-2494cd03b0c9","refre
sh_token":"3a6a26df-f240-3f65-88bd-7dcc5055b934","scope":"api
m:api_publish","token_type":"Bearer","expires_in":3600}
```

4. Type the following cURL command to publish the API. **Modify the Authorization Bearer token with the generated access token,** and "**apiId**" with the id of the GoogleDirectionsAPI which can be found in the json response retrieved when the API is created.

```
curl -k -H "Authorization: Bearer <access token>" -X POST
"https://127.0.0.1:9443/api/am/publisher/v0.14/apis/change-lifecycle?ap
iId=4d0b513e-71d4-489e-9681-31a9178bc189&action=Publish"
```

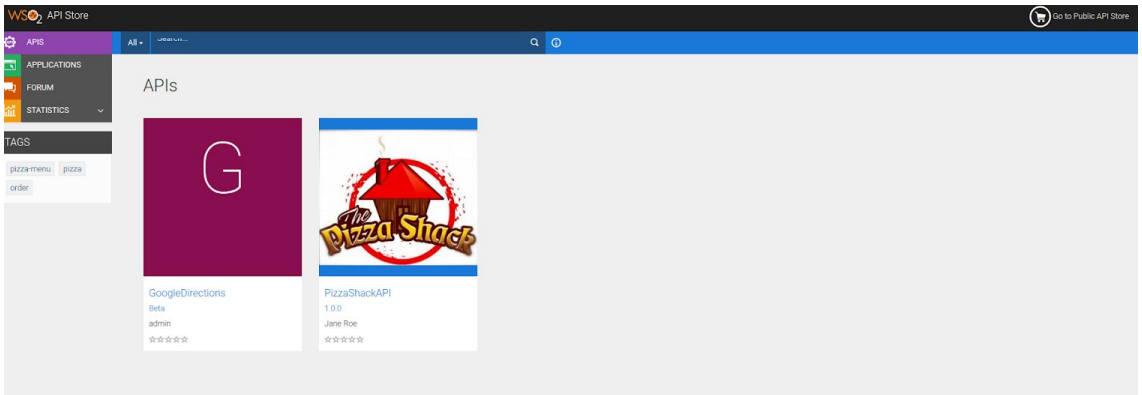
Method : POST
URL:https://127.0.0.1:9443/api/am/publisher/v0.14/apis/change-lifecycle?apiId=**<input apiId here>**&action=Publish
Query Params tab - key 1:apiId, value:<id of api>
Key 2: action, value: Publish
Authorization tab - inherit auth from parent
Header tab - key 1:Authorization, value: Bearer <access token generated previously>

5. Go to the Publisher and refresh. Now the status of the GoogleDirectionsAPI will be displayed as **Published**.



Create New Application

1. Log in as admin to the Store ([https://localhost:9443/store/) and click **carbon.super** on the dashboard. The GoogleDirectionsAPI icon is visible.



2. Open a Command Line Terminal.
3. Modify the [API HOME]/bin/payload.json file clientName to rest_api_store and save;

```
{
    "callbackUrl": "www.google.lk",
    "clientName": "rest_api_store",
    "tokenScope": "Production",
    "owner": "admin",
    "grantType": "password refresh_token",
    "saasApp": true
}
```

4. Open a Command Line Interface and type the following cURL command for client registration.

```
curl -k -X POST -H "Authorization: Basic YWRtaW46YWRtaW4=" -H
"Content-Type: application/json" -d @payload.json
https://localhost:9443/client-registration/v0.14/register
```

Method : Post
URL : https://localhost:9443/client-registration/v0.14/register
Headers tab - Key 1: Content-Type , Value : application/json
Key 2: Authorization, Value : Basic “64encode admin:admin - YWRtaW46YWRtaW4=”
Body tab - paste payload.json code and execute

5. Encode the client ID and secret values as before.
6. Invoke the token endpoint using the following cURL command. Replace the Authorization Basic value.

```
curl -k -d
"grant_type=password&username=admin&password=admin&scope=apim:subs
cribe" -H "Authorization: Basic
ZktWNWFJTXFkclFDRHduV1NMOExvbnRITm84YTpqUm9tUkdWUGhXMnVQZ0Fvd1YzdE
pBVzU5eThh" https://127.0.0.1:8243/token
```

Method : POST
URL : https://127.0.0.1:8243/token
Headers tab - Key 1: Authorization, Value: Basic <encoded value of clientid:clientsecret>
Body tab - Key 1: grant_type, Value: password
Key 2: username, Value: admin, Key 3:password, Value:admin
Key 4: scope, Value: apim:subscribe
Authorization tab - Inherit auth from parent

A response similar to the following is displayed.

```
{"access_token":"9d25befd-36ae-3c8d-828e-445e9e740be2","refre
sh_token":"cb5fb68c-930d-3fe3-bdf1-1654f3fece4a","scope":"api
m:subscribe","token_type":"Bearer","expires_in":3600}
```

7. Remove the content in the [API-M_HOME]/bin/data.json file and add the following:

```
{
    "groupId": "",
    "subscriber": "admin",
    "throttlingTier": "Unlimited",
    "description": "GoogleDiractionsAPI App",
    "status": "APPROVED",
    "name": "GoogleDirectionsAPI"
}
```

8. Run the following cURL command to create an application. Replace the Authorization Bearer value with the access token generated above.

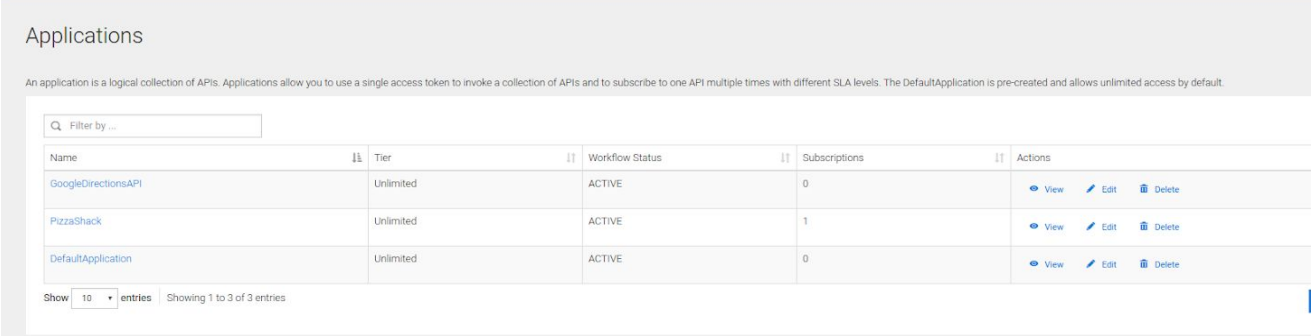
```
curl -k -H "Authorization: Bearer
cbe53aefd029a4a7eaf79817308f4708" -H "Content-Type:
application/json" -X POST -d @data.json
"https://127.0.0.1:9443/api/am/store/v0.14/applications"
```


Method : POST
URL : https://127.0.0.1:9443/api/am/publisher/v0.14/apis
Authorization tab - Type : Inherit auth from parent
Header tab - key 1: Authorization - Bearer <access token>
Key 2: Content-Type - application/json
Body - raw: paste data.json file code

A response similar to the following is displayed.


```
{"applicationId":"ceefbf4c-b221-42c2-bf18-e4faa7dfc42a","name
":"GoogleDirectionsAPI","subscriber":"admin","throttlingTier"
:"Unlimited","callbackUrl":null,"description":"GoogleDiractio
nsAPI
App","tokenType":"OAUTH","status":"APPROVED","groupId":"","ke
ys":[],"attributes":{}}
```


9. The created application for GoogleDirectionsAPI will be listed under **Applications** in the Store.



Create New Subscription

1. Invoke a new authorization token using the following cURL command. Replace the Authorization Basic value with the encoded string.

```
curl -k -d
"grant_type=password&username=admin&password=admin&scope=
apim:subscribe" -H "Authorization: Basic
ZktWNWFJTXFkclFDRHduV1NMOExvbnRITm84YTpqUm9tUkdWUGhXMnVQZ
0Fvd1YzdEpBVzU5eThh" https://127.0.0.1:8243/token
```

Method : POST
URL : https://127.0.0.1:8243/token
Headers tab - Key 1: Authorization, Value: Basic <encoded value of clientid:clientsecret>
Body tab - Key 1: grant_type, Value: password
Key 2: username, Value: admin, Key 3:password, Value:admin
Key 4: scope, Value: apim:subscribe
Authorization tab - Inherit auth from parent

A response similar to the following is displayed.


```
{"access_token":"c5b8debf-7281-3792-8e7b-fb96936fdca9","refre
sh_token":"cb5fb68c-930d-3fe3-bdf1-1654f3fece4a","scope":"api
m:subscribe","token_type":"Bearer","expires_in":3600}
```


2. Retrieve the list of applications using the following cURL command. Replace the Authorization Bearer value.

```
curl -k -H "Authorization: Bearer
c5b8debf-7281-3792-8e7b-fb96936fdca9"
"https://127.0.0.1:9443/api/am/store/v0.14/applications"
```

Method : GET
URL : https://127.0.0.1:9443/api/am/store/v0.14/applications
Authorization tab - Type : Inherit auth from parent
Headers tab - Key 1: Authorization, Value: Bearer <access token>

A response similar to the following is displayed.


```
{"count":3,"next":"","previous":"","list":[{"applicationId":"
3495036c-c5ea-4659-b6ed-22b2bb009585","name":"DefaultApplicat
ion","subscriber":"admin","throttlingTier":"Unlimited","descr
iption":null,"status":"APPROVED","groupId":"","attributes":{}
},{"applicationId":"ceefbf4c-b221-42c2-bf18-e4faa7dfc42a","na
me":"GoogleDirectionsAPI","subscriber":"admin","throttlingTie
r":"Unlimited","description":"GoogleDiractionsAPI
App","status":"APPROVED","groupId":"","attributes":{}},{"appl
icationId":"a221de6f-597d-4c20-be9c-6d92c4ba7ac0","name":"Piz
zaShack","subscriber":"admin","throttlingTier":"Unlimited","d
escription":"","status":"APPROVED","groupId":"","attributes":
{}}]}
```


3. Retrieve the list of APIs using the following cURL command. Replace the Authorization Bearer value.

```
curl -k -H "Authorization: Bearer
c5b8debf-7281-3792-8e7b-fb96936fdca9"
https://127.0.0.1:9443/api/am/store/v0.14/apis
```

Method : GET
URL : https://127.0.0.1:9443/api/am/store/v0.14/apis
Authorization tab - Type : Inherit auth from parent
Headers tab - Key 1: Authorization, Value: Bearer <access token>

A response similar to the following is displayed.


```
{"count":2,"next":"","previous":"","list":[{"id":"f80740db-27
ef-4b39-aa92-c8374993e92d","name":"GoogleDirectionsAPI","desc
ription":"Calculates directions between
locations","context":"/googledirections/Beta","version":"Beta
","provider":"admin","status":"PUBLISHED","thumbnailUri":null
,"scopes":[]},{"id":"eb43b898-f5fb-423c-b21c-5c3050c380d0","n
ame":"PizzaShack","description":null,"context":"/pizzashack/1
.0.0","version":"1.0.0","provider":"admin","status":"PUBLISHE
D","thumbnailUri":"/apis/eb43b898-f5fb-423c-b21c-5c3050c380d0
/thumbnail","scopes":[{"key":null,"name":"order_pizza","roles
":[]},{"key":null,"name":"order_pizza","roles":[]}]}],"pagina
tion":{"total":2,"offset":0,"limit":25}}
```


4. Replace the text in the [API-M_HOME]/bin/data.json file with the following. Give the retrieved apiIdentifier and applicationId.

```
{
    "tier": "Gold",
    "apiIdentifier": "4d0b513e-71d4-489e-9681-31a9178bc189",
    "applicationId": "645c9838-7b74-4fd1-8b5a-2252909a0342"
}
```

5. Type the following cURL command to create a new subscription replacing the Authorization Bearer value.

```
curl -k -H "Authorization: Bearer d1a167e580e40a8a2ae297fc4656677c" -H
"Content-Type: application/json" -X POST  -d @data.json
"https://127.0.0.1:9443/api/am/store/v0.14/subscriptions"
```

Method : POST
URL : https://127.0.0.1:9443/api/am/store/v0.14/subscriptions
Authorization tab - Type : Inherit auth from parent
Header tab - key 1: Authorization - Bearer <access token>
Key 2: Content-Type - application/json
Body - raw: paste new data.json file code

A response similar to the following is displayed.


```
{"subscriptionId":"cf3cfd3a-68d8-4141-acb1-3664254c97b9","app
licationId":"a221de6f-597d-4c20-be9c-6d92c4ba7ac0","apiIdenti
fier":"admin-GoogleDirectionsAPI-Beta","tier":"Gold","status"
:"UNBLOCKED"}
```


6. Go to the Store and click **APPLICATIONS.**

7. Select the GoogleDirectionsAPI from the list. Select the **Subscriptions** tab. There will be a subscription tier field with a Gold Subscription.