



# WSO2 API Manager 2.2.0

**Labkit**

Developer Fundamentals

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

**Tel:** +1 408 754 7388 | **Fax:** +1 408 689 4328 | **Email:** [training@wso2.com](mailto:training@wso2.com)

## Table of Contents

[Lab: Getting Started with WSO2 API Manager Training Objective](#)

[Lab: Defining Users and Roles](#)

[Lab: Creating and Publishing the PizzaShack API](#)

[Lab : Import/Export API](#)

[Lab: Working with Tenants](#)

[Lab: Subscribing to APIs](#)

[Lab: Invoking the API](#)

[Lab: Working with Throttling Policies](#)

[Lab: Analyze Runtime Statistics](#)

[Lab: Managing Alerts with Real-Time Analytics](#)

[Lab: Using Published APIs](#)

## 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
- Deploy and test JAX-RS service

## 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] <https://docs.wso2.com/display/AM220/Installation+Prerequisites>

[2] <https://docs.wso2.com/display/AM220/Installing+on+Linux+or+OS+X>

[3] <https://docs.wso2.com/display/AM220/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] <https://docs.wso2.com/display/AM220/Analytics>

## Other Installations

In order to complete the use case described in this labkit the following products must be installed:

Installing brew[1], JDBC driver for MySQL[2], cURL[3], Google Chrome [4], Advanced Rest Client [5],

[1] Install brew from <https://brew.sh/>

[2] In terminal run

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

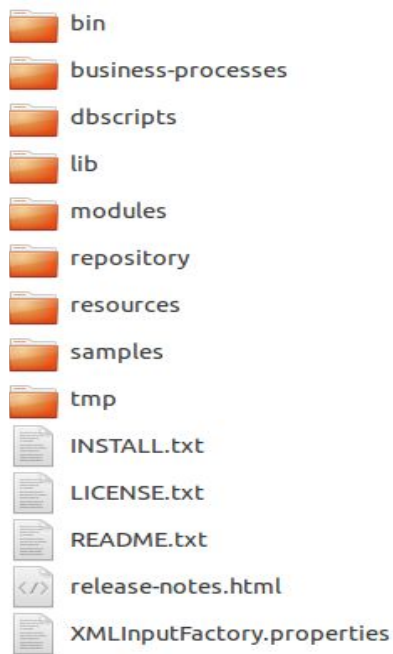
[3] In terminal run: 'brew install curl'

[4] <https://support.google.com/chrome/answer/95346?hl=en>

[5] <https://chrome.google.com/webstore/detail/advanced-rest-client/hgmloofddfdnphfgcellkdfbfbjeloo>

## 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 Management 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 kind 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
<code>&lt;API-M_HOME&gt;/repository/conf/carbon.xml</code>	0
<code>&lt;Analytics_HOME&gt;/repository/conf/carbon.xml</code>	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] <https://docs.wso2.com/display/AM220/Running+the+Product>

## 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

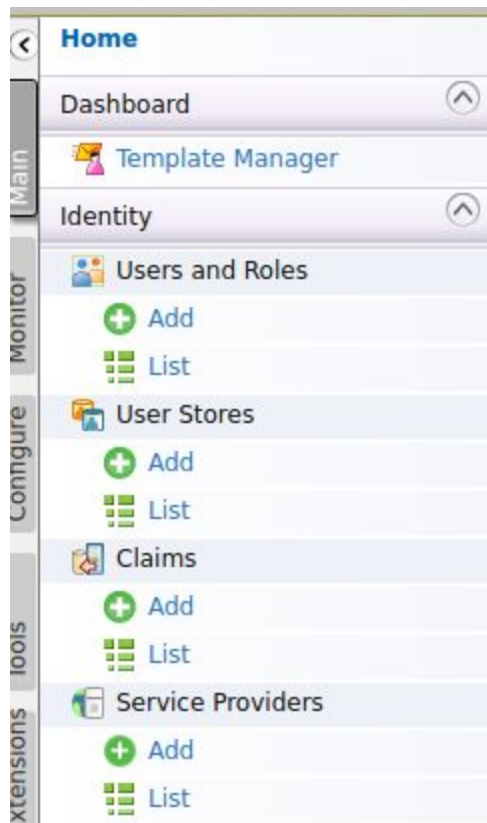
- Define roles
- Define users via the admin console



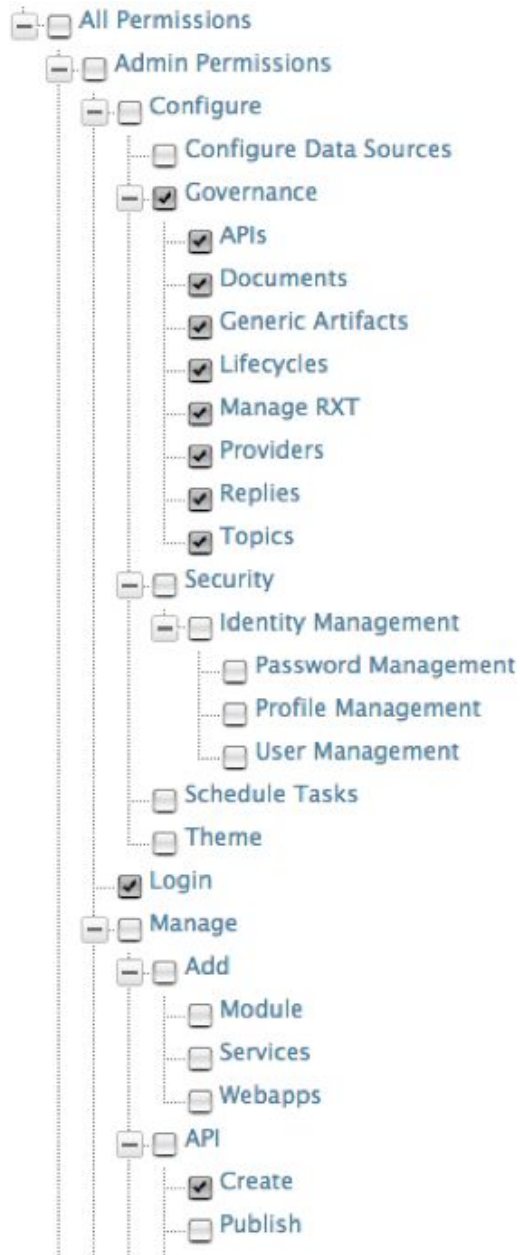
## Detailed Instructions

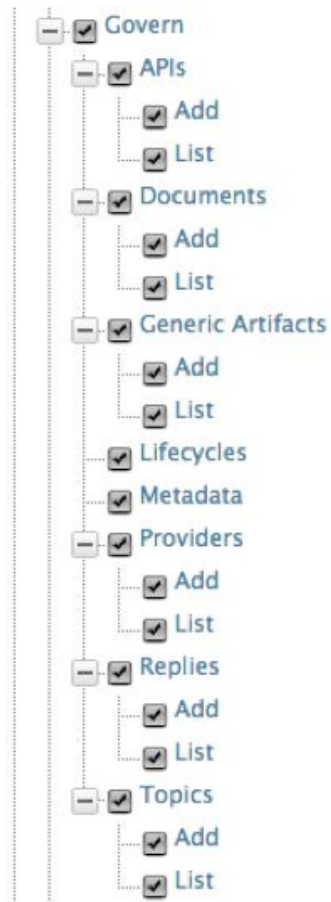
### Define Roles

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://hostname: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 **creator** 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 **publisher** role, with the **following permissions**:

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

## 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**
2. Click **Add New User**
3. Provide user name (apicreator) and password (password)
4. Click **Next**.
5. Select the creator role.
6. Click **Finish**.

**Repeat the steps** to create a user (apipublisher) in the publisher role.

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.

## Lab: Creating and Publishing the PizzaShack API

### 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 PizzaShack API to the store
- Implement APIs
- Manage APIs
- Add documentation
- Publish the APIs

### Detailed Instructions

#### Add the PizzaShack API to the Store

Now that we set up the API-M and added users, we are ready to publish the API the Pizza Shack 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	<b>PizzaAPI</b>	Name of API as you want it to appear in the API store
Context	<b>pizzashack</b>	URI context path that is used by API consumers (Application Developers)
Version	<b>1.0.0</b>	API version (in the form of version.major.minor)
Visibility	Public	Whether this API is visible to all or restricted to certain roles.
Thumbnail Image	Download a PizzaShack logo image and upload it	Icon to be displayed in API store (can be jpeg, tiff, png format) - Under <b>Advanced Options</b> .
Description	Pizza API: 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
API Definition		<p>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.</p> <p>API resources can accept following optional attributes:</p> <ul style="list-style-type: none"> <li>• <b>verbs:</b> Specifies the HTTP verbs a particular resource would accept. Allowed values are GET, POST, PUT, DELETE. Multiple values can be specified.</li> <li>• <b>uri-template:</b> A URI template as defined in <a href="http://tools.ietf.org/html/rfc6570">http://tools.ietf.org/html/rfc6570</a> (eg: /phoneverify/{phoneNumber})</li> <li>• <b>url-mapping:</b> A URL mapping as defined as per the servlet specification (extension mappings, path mappings and exact mappings)</li> </ul>

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

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

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

URL Pattern: /pizzashack/1.0.0 Uri Pattern E.g.: path/to/resource

☐ GET ☐ POST ☐ PUT ☐ DELETE ☐ PATCH ☐ HEAD more

**Add**

POST	/order	+ Summary	
GET	/menu	+ Summary	
PUT	/order/{orderid}	+ Summary	
GET	/order/{orderid}	+ Summary	

**Save** **Next: Implement >**

8. Click **Implement**.

## Implement APIs

1. Select **Managed API**.

**1 Design** **2 Implement** **3 Manage**

**Managed API**  
Provide the production and sandbox endpoints of the API to be managed.

**Prototyped API**  
Use the inbuilt JavaScript engine to prototype the API or provide an endpoint to a prototype API. The inbuilt JavaScript engine does not have support to prototype SOAP APIs

## 2. Specify the following.

Field	Value	Description
Endpoint Type	HTTP/REST Endpoint	
Production Endpoint	<a href="https://localhost:9443/am/sample/pizzas/hack/v1/api/">https://localhost:9443/am/sample/pizzas/hack/v1/api/</a>	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.
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



## 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

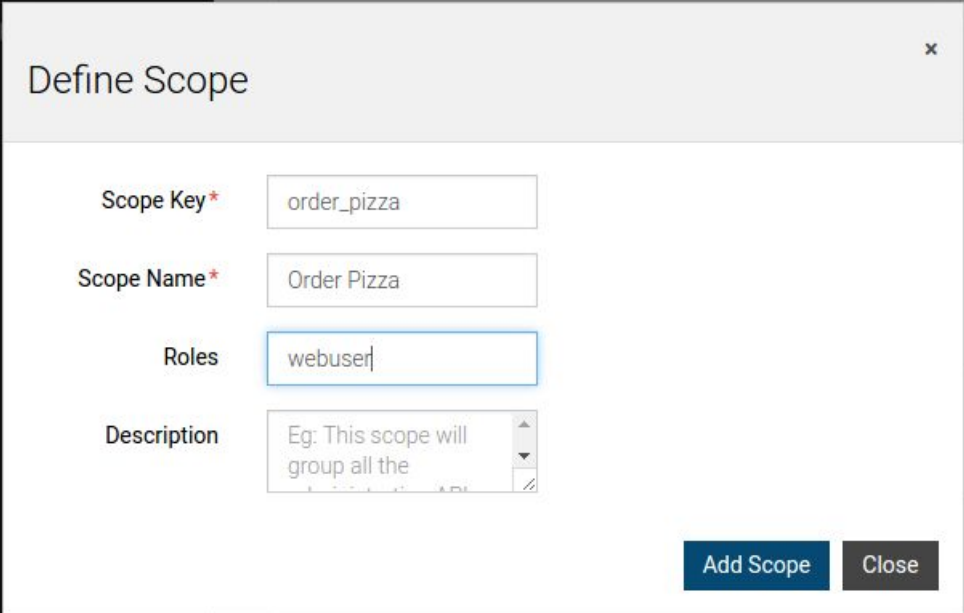
For this use case, we will also be making use of OAuth2.0 scopes. Therefore we will be creating a scope named `order_pizza` and allowing that scope to only users with `webuser` role.

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**.



**Define Scope**


Scope Key \*

Scope Name \*

Roles

Description

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

Scopes: **order\_pizza** Order Pizza 

Roles : webuser

POST	/order	<a href="#">+ Summary</a>	Application & Application User	Unlimited	<b>Order Pizza</b>
GET	/menu	<a href="#">+ Summary</a>	Application & Application User	Unlimited	<a href="#">+ Scope</a>
PUT	/order/{orderId}	<a href="#">+ Summary</a>	Application & Application User	Unlimited	<a href="#">+ Scope</a>
GET	/order/{orderId}	<a href="#">+ Summary</a>	Application & Application User	Unlimited	<b>Order Pizza</b>

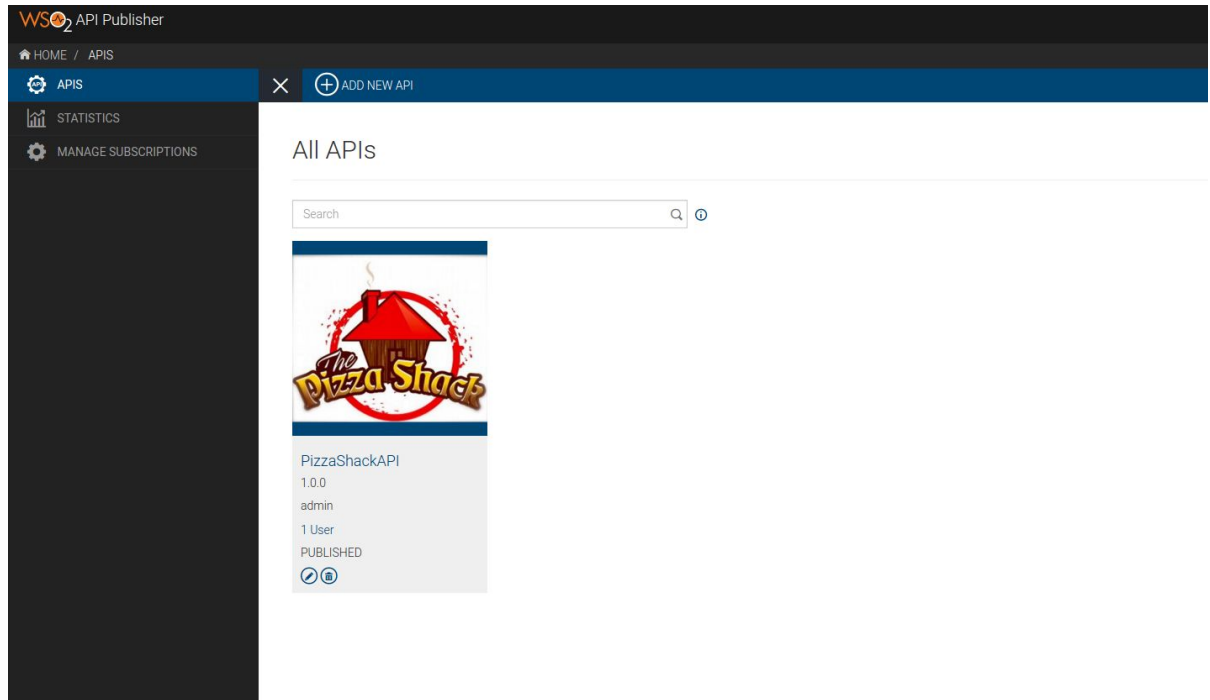
**Note:** The order in which the scopes are shown in the image above can differ from the order on screen. Make sure you add the scopes on the POST/order and GET/order{orderId}.

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.

- Click **Save**.

## 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:

## PizzaShackAPI - 1.0.0

[Overview](#)
[Lifecycle](#)
[Versions](#)
[Docs](#)
[Users](#)



0 Users

PUBLISHED

Docs

View in Store

Description

This document describe a RESTful API for Pizza Shack online pizza delivery store.

Visibility	Public
Context	/pizzashack/1.0.0
Production URL	https://localhost:9443/am/sample/pizzashack/v1/api/
Sandbox URL	https://localhost:9443/am/sample/pizzashack/v1/api/
Date Last Updated	8/4/2016, 1:51:09 PM
Tier Availability	Unlimited
Default API Version	None
Tags	order,pizza-menu,pizza
Business Owner	Jane Roe [marketing@pizzashack.com]
Technical Owner	John Doe [architecture@pizzashack.com]
Published Environments	Production and Sandbox

- 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:

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

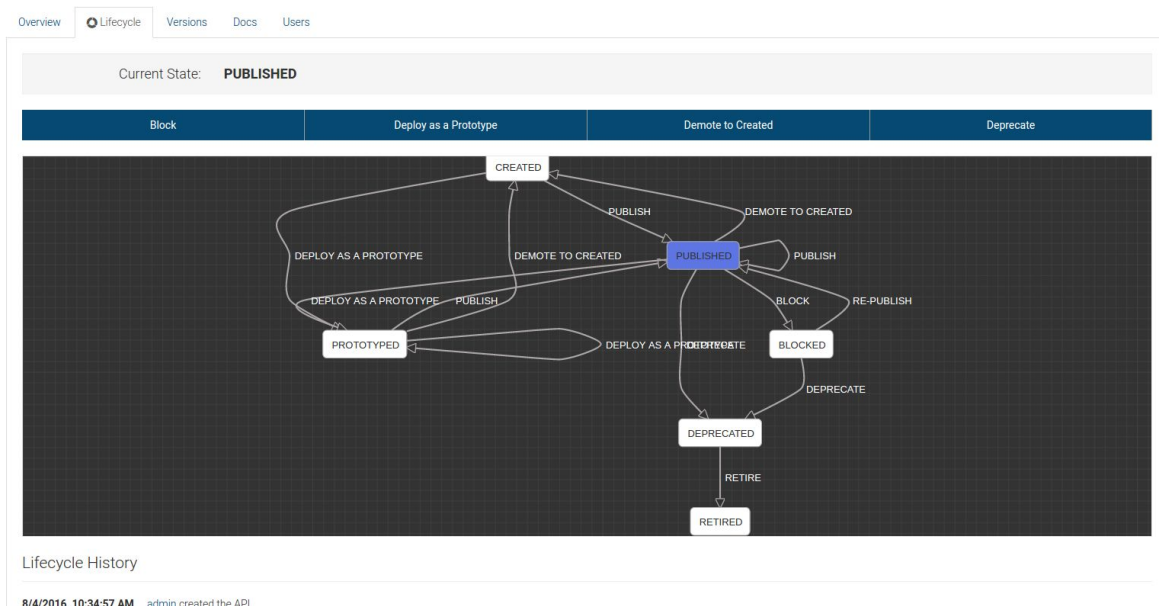
## Publish the API

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 PizzaShack API 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 PizzaShack API (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
- Export PizzaShack API as zip file
- Import API using the API Import Export Tool

### Export and Import an API

1. Download the API Import Export tool at [link](#).
2. Copy the downloaded file to  
`<API-M_HOME>/repository/deployment/server/webapps`

### Exporting an API

1. Export the API by executing the following CURL in a command line interface
 

```
curl -H "Authorization:Basic <token>"
-X GET
"https://<APIM_HOST:Port>/api-import-export-<version>/export-api?name=<API-name>&version=<API-version>
&provider=<API-provider>" -k > <exportedApiFileName>.zip
```

The Basic Auth token should be base64 encoded credentials separated by a colon

e.g., Base64 Encoded Value of **admin:admin**

Go to <https://www.base64decode.org/> and encode the credentials

### Sample CURL Command

```
curl -H "Authorization:Basic YWRtaW46YWRtaW4=" -X GET
"https://localhost:9443/api-import-export-2.2.0-v2/export-api?name=PizzaShackAPI&version=1.0.0&provider=apicreator" -k >
PizzaShack_API.zip
```

2. Once execution is completed you will see the following result in your terminal and a new file with the name PizzaShack\_API.zip. (A successful export will show a 'k' behind the Average DLoad and Current Speed values)

```
% Total    % Received % Xferd  Average Speed   Time    Time       Time  Current
   100    147k    0   147k    0     0    657k      0  --:--:--  --:--:--  --:--:--   654k
```

### Importing an API

1. Download the WeatherAPI.zip file from <https://github.com/wso2/WSO2-Training/releases/download/APIM210DF/WeatherAPI.zip>. Import the Weather API using following command

```
curl -H "Authorization:Basic YWRtaW46YWRtaW4=" -F
file=@"WeatherAPI.zip" -k -X POST
"https://localhost:9443/api-import-export-2.2.0-v2/import-api"
```

2. Log into the Publisher. You will see the imported Weather API is in Created state. To make this API available in the Store you need to publish the API.

### Expected Outcome

The Pizzashack API was exported and the Weather API was imported successfully using the API Import Export Tool



## 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.

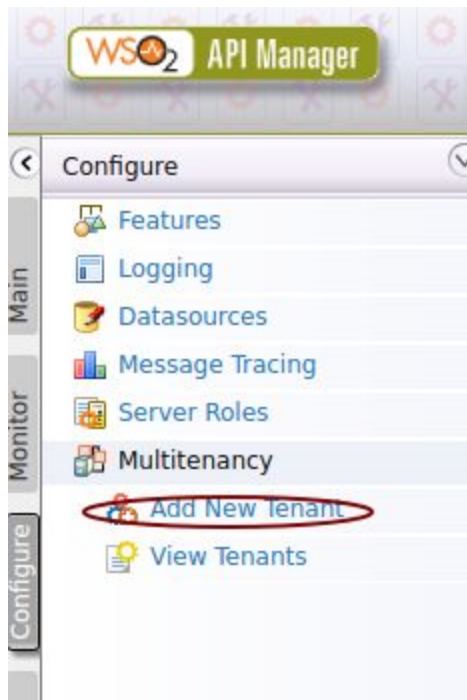
### High Level Steps

- 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:

Home > Configure > Multitenancy > Add New Tenant Help

### Register A New Organization

**Domain Information**

Domain \*   
Use a domain for your organization, in the format "example.com". This domain should be unique.

**Usage Plan Information**

Select Usage Plan For Tenant\*  ⌵  
According to the selected plan, resources will be allocated to you. You can update or downgrade your plan later according to your requirements.

**Tenant Admin**

First Name\*   
 Last Name\*   
 Admin Username \*  @pizzashack.com  
 Admin Password \*   
 Admin Password (Repeat) \*

**Contact Details**


Email\*

---

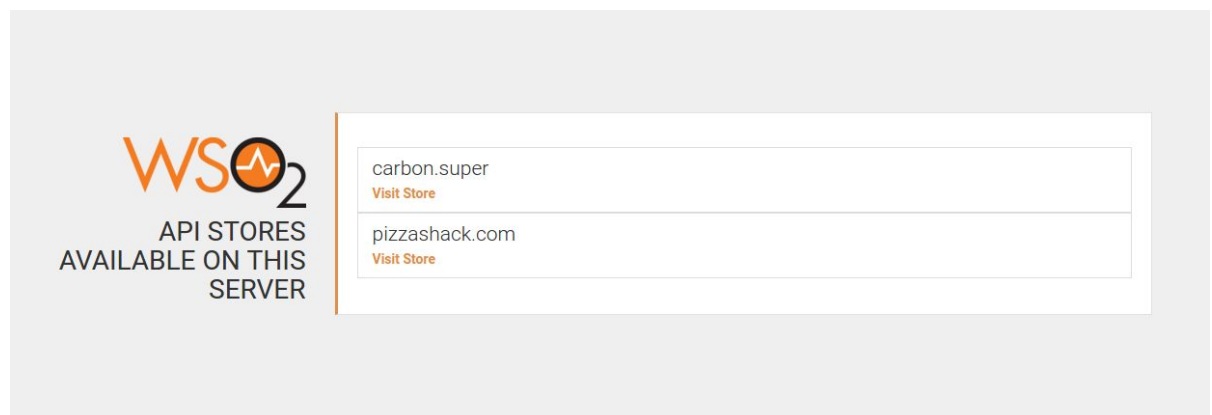
Enter the Tenant Domain:   Help

### Tenants List

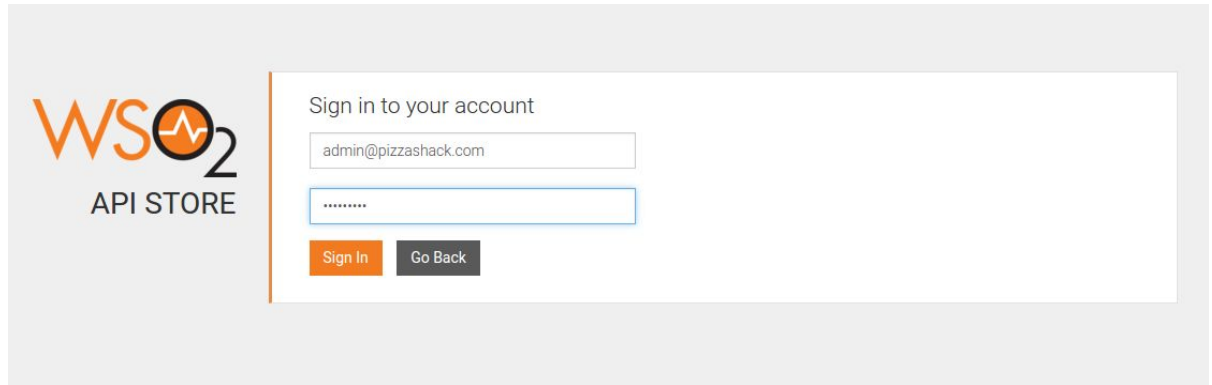
Domain	Email	Created Date	Active	Edit
pizzashack.com		5/16/13 16:13:13	<input checked="" type="checkbox"/>	<input type="button" value="Edit"/>

 You have registered the Organization Successfully

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

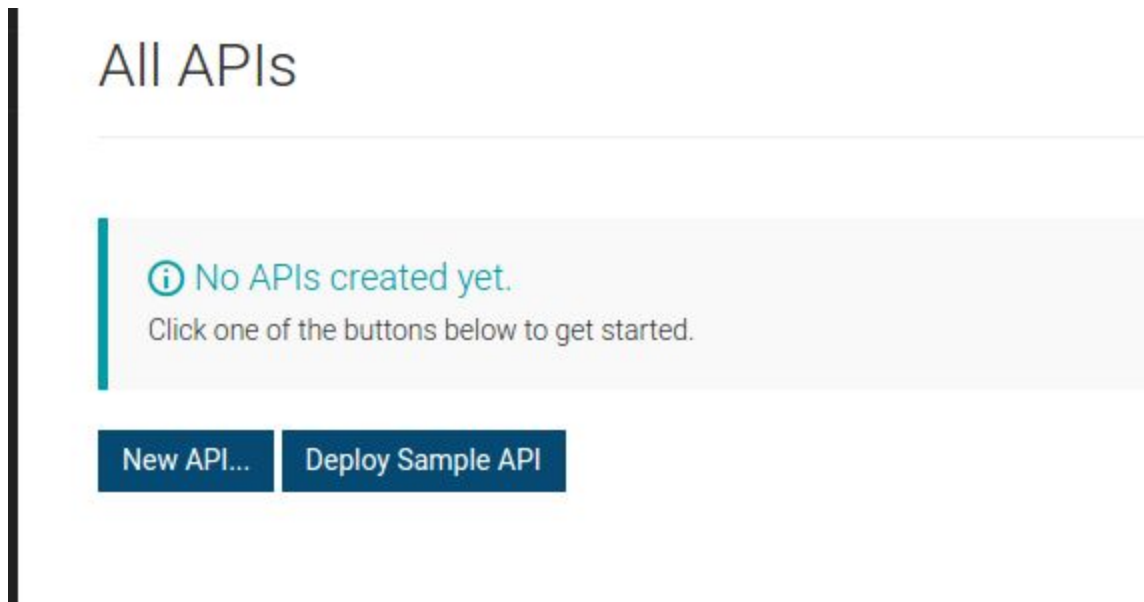


5. Click on pizzashack.com and note that no APIs have been published yet.  
 6. Log in as admin@pizzashack.com. There are no APIs displayed even after logging in.

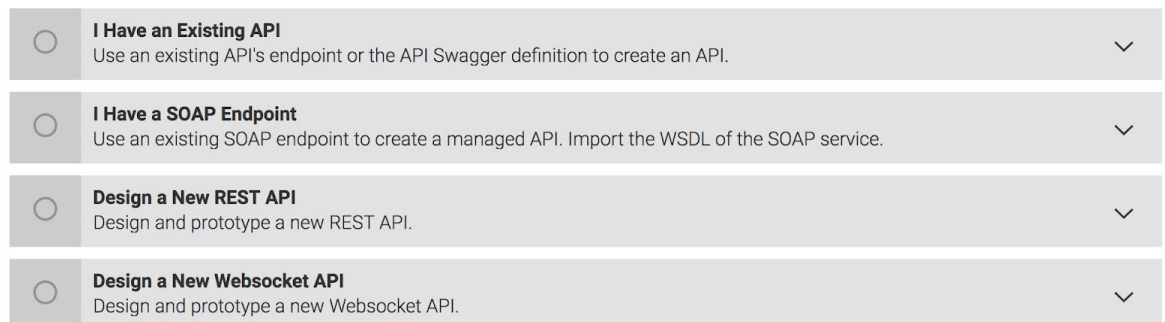


## Manage the API within Tenant

1. Log in to the Publisher as [admin@pizzashack.com](mailto:admin@pizzashack.com).
2. Click **New API....**



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



## 4. Enter the following details.

**1 Design** **2 Implement**

**General Details**

Name: \* ? CustomerInfo

Context: \* ? customer

Version: \* 1.0.0

Visibility: ? Visible to my domain

Description:   
Maximum 20000 characters.

Tags: ? Add tags  
Type a Tag and Enter

## 5. Add the following resources.

Resource URL	Methods
CreateCustomer	POST
QueryCustomerInfo	GET
UpdateCustomerInfo	PUT
DeleteOrderInfo	DELETE

**API Definition**

URL Pattern: customer/1.0.0 Url Pattern E.g.: path/to/resource Import Edit Source

☐ GET ☐ POST ☐ PUT ☐ DELETE ☐ PATCH ☐ HEAD more

+ Add

POST	/CreateCustomer	<a href="#">+ Summary</a>	<span>ⓘ</span>
GET	/QueryCustomerInfo	<a href="#">+ Summary</a>	<span>ⓘ</span>
PUT	/UpdateCustomerInfo	<a href="#">+ Summary</a>	<span>ⓘ</span>
DELETE	/DeleteOrderInfo	<a href="#">+ Summary</a>	<span>ⓘ</span>

Save Next: Implement >

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

**POST** /CreateCustomer [+ Summary](#)

Description : [+ Add Implementation Notes](#)

Produces : [Empty](#) Consumes : [Empty](#)

Parameters :

Parameter Name	Description	Parameter Type	Data Type	Required	Delete
Payload	<a href="#">Request Body</a>	<a href="#">body</a>	<a href="#">+ Empty</a>	<a href="#">True</a>	<a href="#">Delete</a>

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

**GET** /QueryCustomerInfo [+ Summary](#)

Description : [+ Add Implementation Notes](#)

Produces : [Empty](#) Consumes : [Empty](#)

Parameters :

Parameter Name	Description	Parameter Type	Data Type	Required	Delete
MobileNumber	<a href="#">The caller's phone number</a>	<a href="#">query</a>	<a href="#">string</a>	<a href="#">True</a>	<a href="#">Delete</a>

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

**PUT** /UpdateCustomerInfo [+ Summary](#)

Description : [+ Add Implementation Notes](#)

Produces : [Empty](#) Consumes : [Empty](#)

Parameters :

Parameter Name	Description	Parameter Type	Data Type	Required	Delete
Payload	<a href="#">Request Body</a>	<a href="#">body</a>	<a href="#">+ Empty</a>	<a href="#">True</a>	<a href="#">Delete</a>

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

**DELETE** /DeleteOrderInfo [+ Summary](#)

Description : [+ Add Implementation Notes](#)

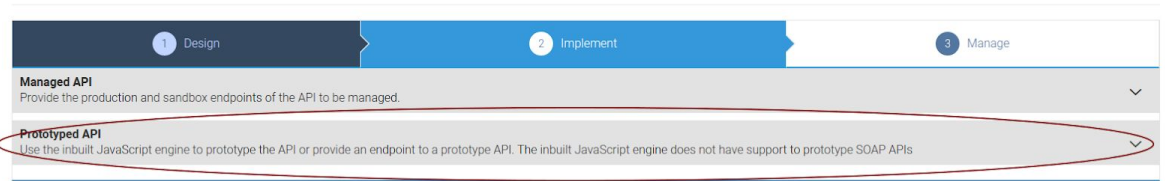
Produces : [Empty](#) Consumes : [Empty](#)

Parameters :

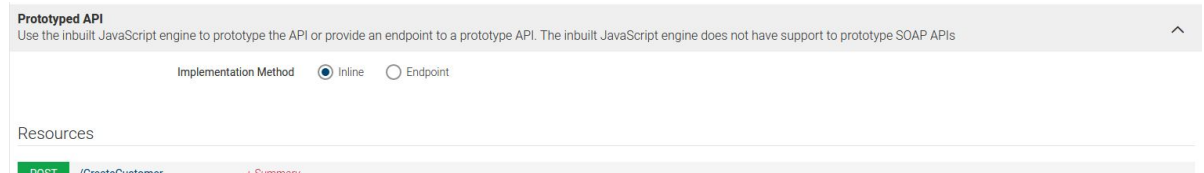
Parameter Name	Description	Parameter Type	Data Type	Required	Delete
orderid	<a href="#">The order ID that should be deleted</a>	<a href="#">query</a>	<a href="#">string</a>	<a href="#">True</a>	<a href="#">Delete</a>

9. Click **Implement**.
10. Click **Prototyped API**.

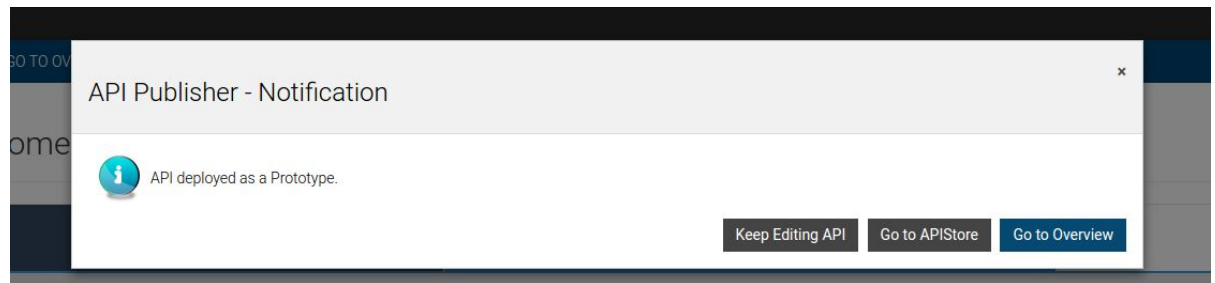
CustomerInfo: /t/pizzashack.com/customer/1.0.0



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



12. Click **Deploy as Prototype**.



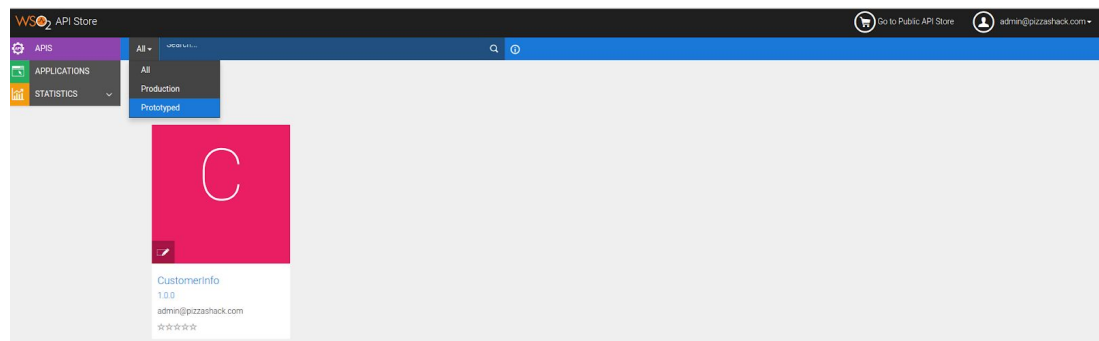
13. Click **Go to Overview** and to see the overview of the newly created API.

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 PizzaShack APIs as a base can open the API store and check its contents and subscribe to the API if interested.

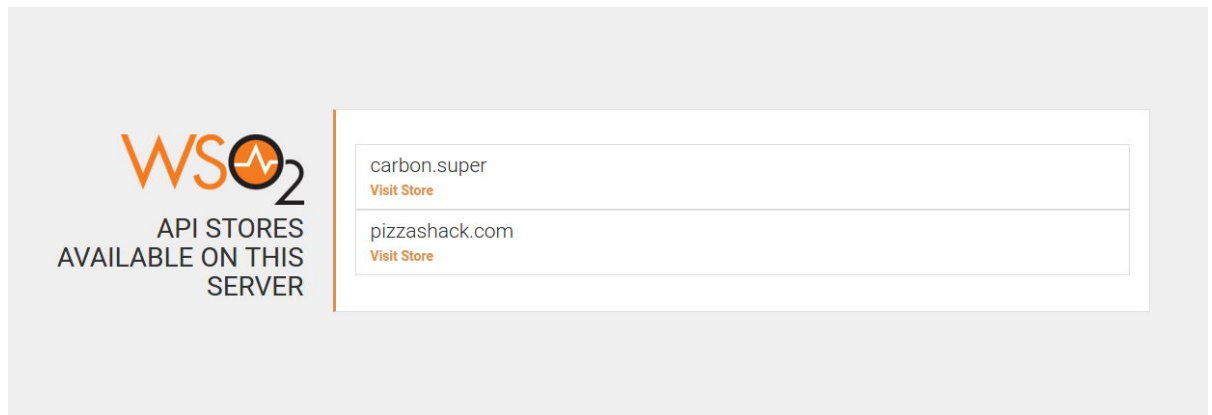
### High Level Steps

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

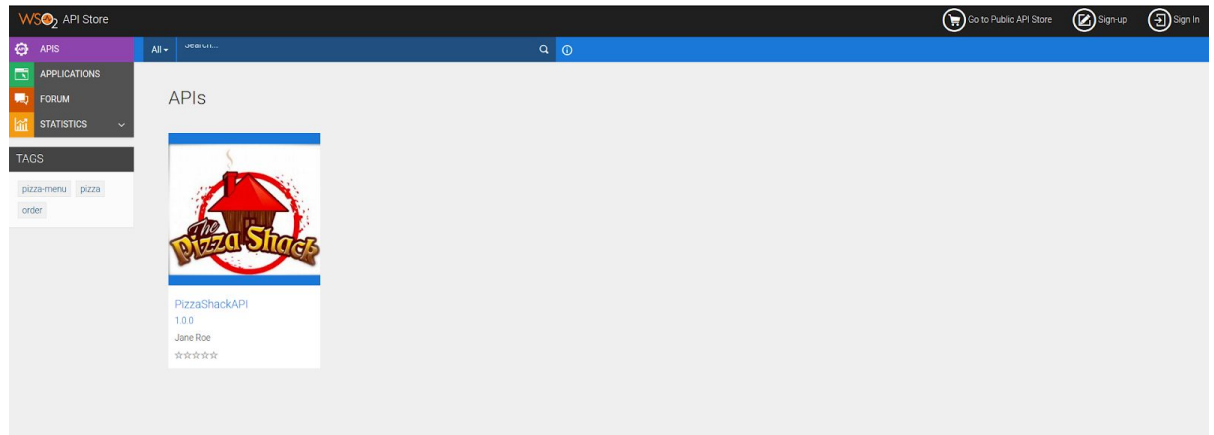
### Detailed Instructions

#### Browse the Store

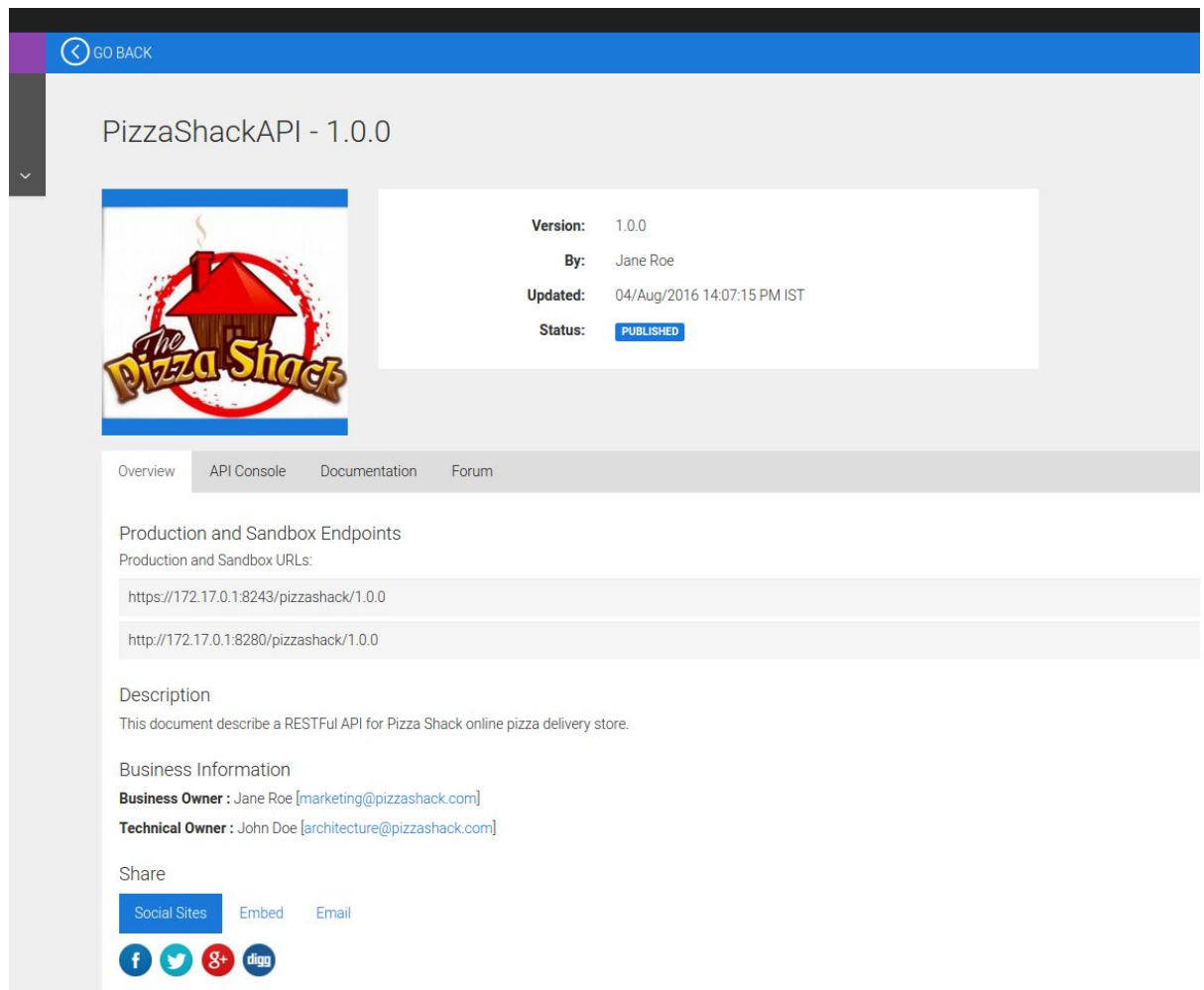
1. To view the API store contents, open the following URL: <https://localhost:9443/store>.



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



3. 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.

**Add Application**

An application is a logical collection of APIs. Applications allow you to use a single access token to invoke a collection of APIs and to subscribe to one API by default.

**Name\***  Characters left: 60

**Per Token Quota**  Allows unlimited requests

**Description**

- 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).

PizzaShackAPI - 1.0.0

**Version:** 1.0.0  
**By:** Jane Roe  
**Updated:** 04/Aug/2016 14:07:15 PM IST  
**Status:** PUBLISHED  
**Rating:** ☆☆☆☆

**Applications**

**Tiers**

Overview API Console Documentation Forum

Production and Sandbox Endpoints  
 Production and Sandbox URLs:  
<https://172.17.0.1:8243/pizzashack/1.0.0>  
<http://172.17.0.1:8280/pizzashack/1.0.0>

- Click **Subscribe**.
- Switch to the **APPLICATIONS** page. Select PizzaShack application from the list.
- Click the **Production Keys** tab.
- 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).

The screenshot shows the 'PizzaShack' application configuration page in the API Manager. The left sidebar contains navigation links for APPLICATIONS, APIS, FORUM, and ANALYTICS. The top navigation bar includes APPLICATION LIST and EDIT. The main content area has tabs for Details, Production Keys, Sandbox Keys, and Subscriptions. Under the 'Details' tab, there is a message 'No Keys Found' and a 'Grant Types' section. The 'Grant Types' section lists several options: Refresh Token, IWA-NTLM, SAML2, Client Credential, Implicit, Code, and Password. The 'Access token validity period' is set to -1 seconds. A 'Generate keys' button is at the bottom.

**APPLICATIONS** APPLICATION LIST EDIT

**PizzaShack**

Details Production Keys Sandbox Keys Subscriptions

**No Keys Found**  
No keys are generated for this type in this application.

**Grant Types**  
The application can use the following grant types to generate Access Tokens. Based on the application requirement, you can enable or disable grant types for this application.

☒ Refresh Token ☒ SAML2 ☐ Implicit ☒ Password  
☒ IWA-NTLM ☒ Client Credential ☐ Code

**Callback URL**

**Scopes**

**Access token validity period**  
 Seconds

**Generate keys**

**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`.

You have now successfully subscribed to the API and can start using it.

## 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 vMxNW6ILwNrWvnKJyewejSIHZFka

Using cURL, this is very simple - Let's exercise the Menu API.

1. Open a new Command Line Interface.
2. Type

```
curl -v http://localhost:8280/pizzashack/1.0.0/menu
```

You will see the following message:

```
* 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>
```

- 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"}]

```

## 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 PizzaAPI 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.

Name	Quota Policy	Quota	Unit Time	Rate Limit	Time Unit	
Bronze	requestCount	1000	1 min	NA	NA	<a href="#">Edit</a> <a href="#">Delete</a>
Gold	requestCount	5000	1 min	NA	NA	<a href="#">Edit</a> <a href="#">Delete</a>
Platinum	requestCount	25	1 min	5	sec	<a href="#">Edit</a> <a href="#">Delete</a>
Silver	requestCount	2000	1 min	NA	NA	<a href="#">Edit</a> <a href="#">Delete</a>

3. Enter the following information

## Add Subscription Throttle Policy

### General Details

Name \*

Description

### Quota Limits

☒ Request Count ☐ Request Bandwidth

Request Count \*

Unit Time \*

### Burst Control (Rate Limiting)


Request Count

### Policy Flags

Stop On Quota Reach ☒

Billing Plan

### Custom Attributes

 Add Custom Attribute

### Permissions

Roles \*

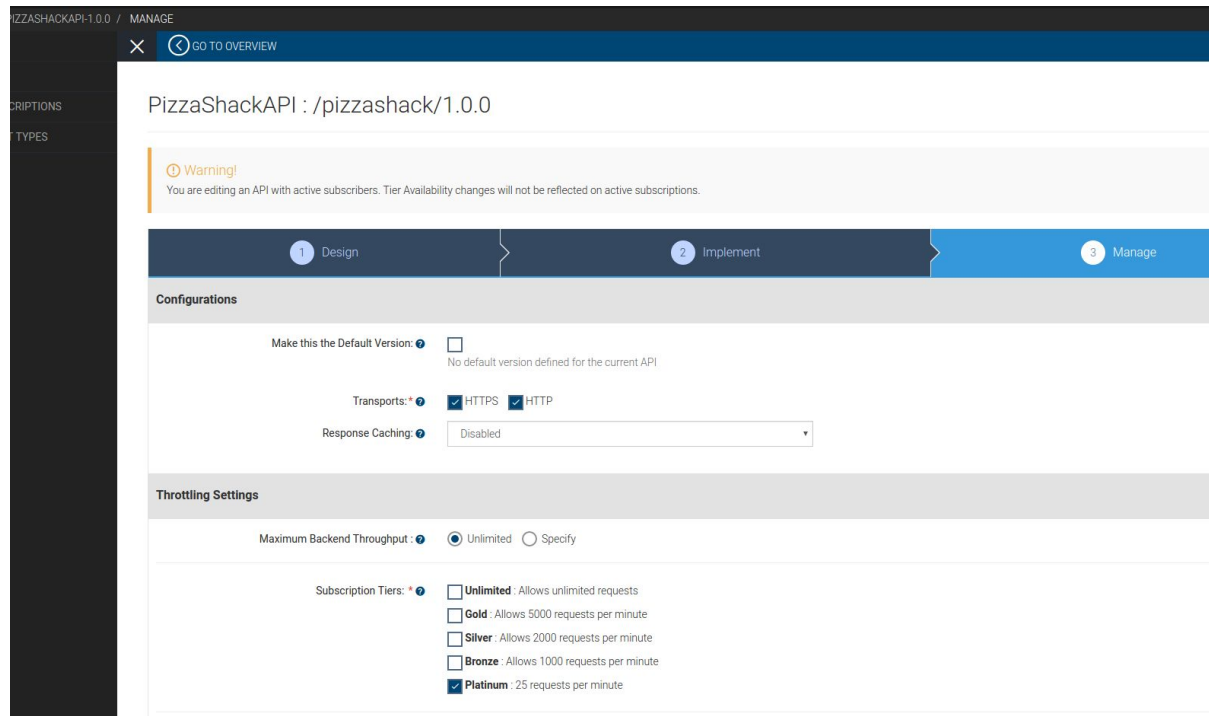
This tier is **Allowed** for above roles.

☒ Allow ☐ Deny

 Save  Cancel

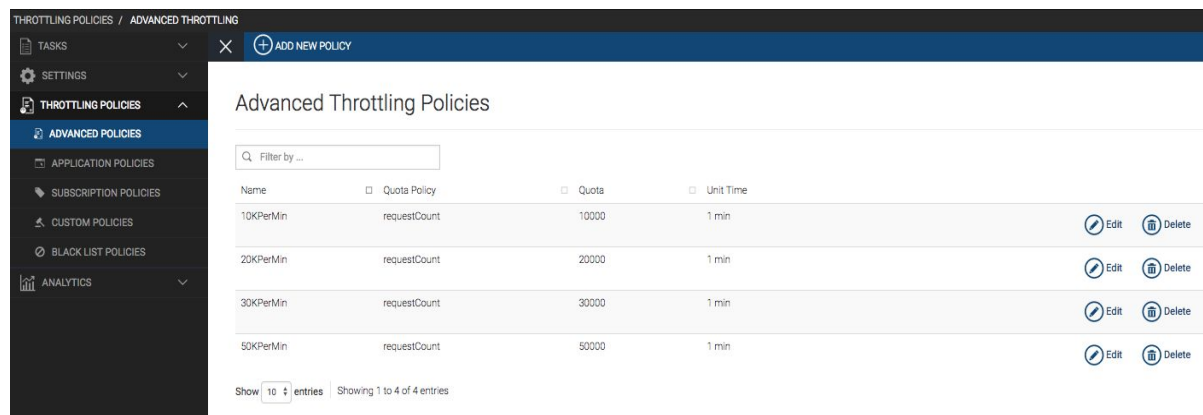
In the API **Publisher**, edit the PizzaAPI and note that Platinum can now be selected under **Subscription Tiers** in the **THROTTLING SETTINGS** section, edit and save it, which then will be visible in the API Store whenever a person wants to subscribe to the PizzaAPI





## 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.

## Add Advanced Throttle Policy

### General Details

Name: \*

Description:


### Default Limits

☒ Request Count ☐ Request Bandwidth

Request Count: \*

Unit Time: \*  Minutes(s)


### Conditional Groups


 Add Conditional Group


0


Condition Group

Sample description about condition group


 IP Condition
 ☐

 Header Condition
 ☒

 Query Param Condition
 ☐

 JWT Claim Condition
 ☐

Header Condition Policy

On 

This configuration is used to throttle based on Headers.

Header Name: \*

Param Value: \*

Add

Invert Condition:

☐

### Execution Policy

Request Count : Request Count

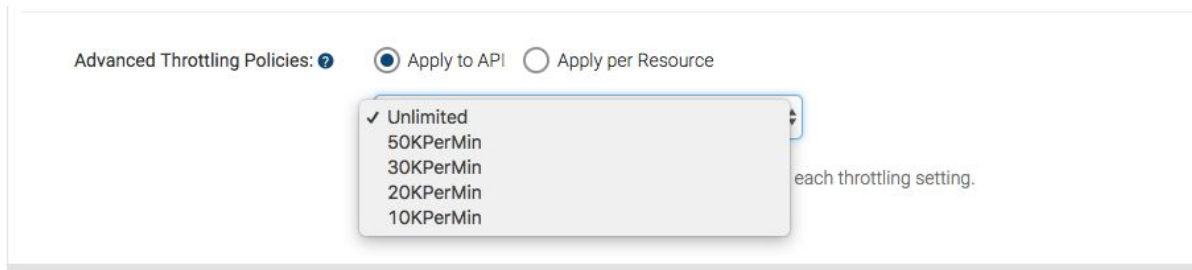
Request Count: \*

Time: \*  Minute(s)

 Save  Cancel

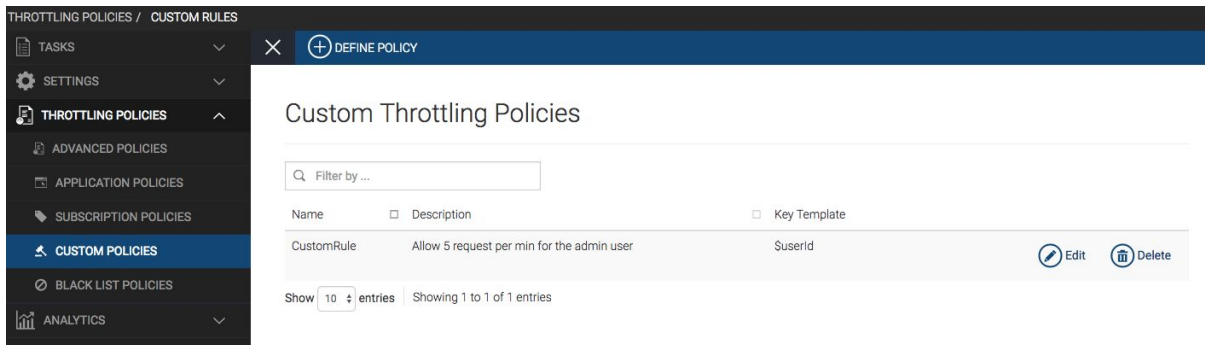
- Click **Add** to add the Header condition Policy and Click **Save**.

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

**WSO2 Admin Portal**

**GO BACK**

### Edit Custom Policy

**Name \*** CustomRule

**Description** Allow 5 requests per minute for the admin user

**Key Template \*** \$userid

**Siddhi 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;
```

**Hide Sample**

Following query will allow 5 request per minute for Admin user  
 Key Template: **\$userid**  
 Siddhi 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;

**Apply Rule** **Cancel**

WSO2 API Manager | © 2017 WSO2, Inc.

Add following details.

<b>Name</b>	CustomPolicy
<b>Description</b>	Allow 5 requests per minute for admin user
<b>Key Template</b>	\$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 **PizzaAPI**.
2. Click **Manage** and make sure the **Apply per Resource** is selected under the **Advanced Throttling Policies**.

Advanced Throttling Policies: ☒ Apply to API ☐ Apply per Resource

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).



Create your Account

Username \*

Characters left: 25

Password \*

Re-type Password \*

First Name \*

Last Name \*

Email \*

[View Additional Details](#)

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

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 bb79f466-35fd-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": "Spinach 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 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

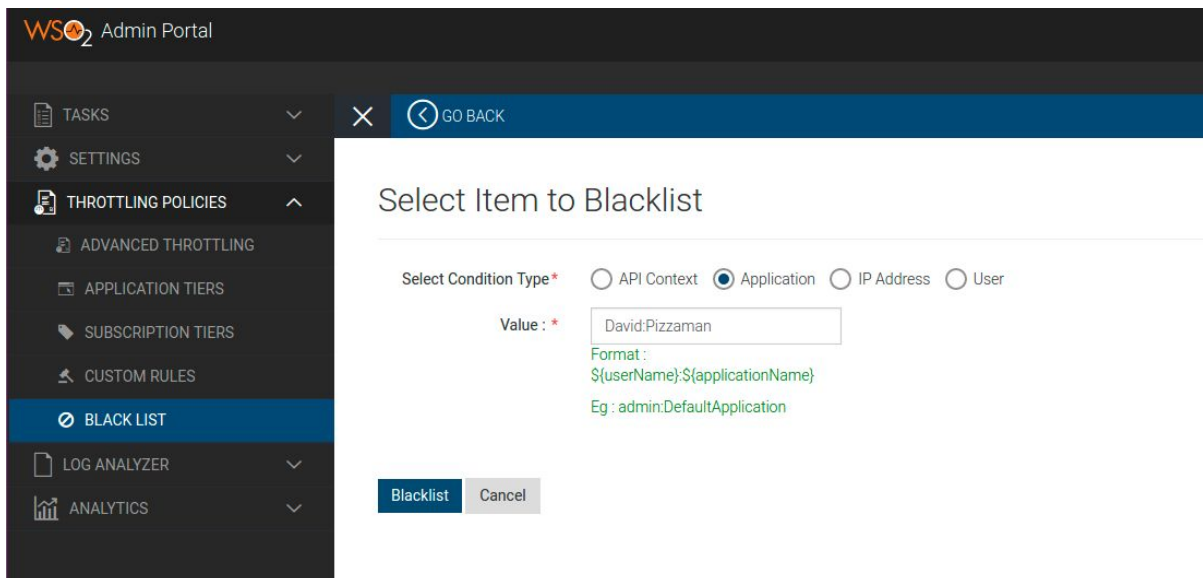
Alternatively you can use the following CURL command.

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'
```

Application PizzaMan can now successfully invoke the API.

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



WSO2 Admin Portal

TASKS

SETTINGS

THROTTLING POLICIES

ADVANCED THROTTLING

APPLICATION TIERS

SUBSCRIPTION TIERS

CUSTOM RULES

BLACK LIST

LOG ANALYZER

ANALYTICS

GO BACK

### Select Item to Blacklist

Select Condition Type\*

☐ API Context ☒ Application ☐ IP Address ☐ User

Value : \*

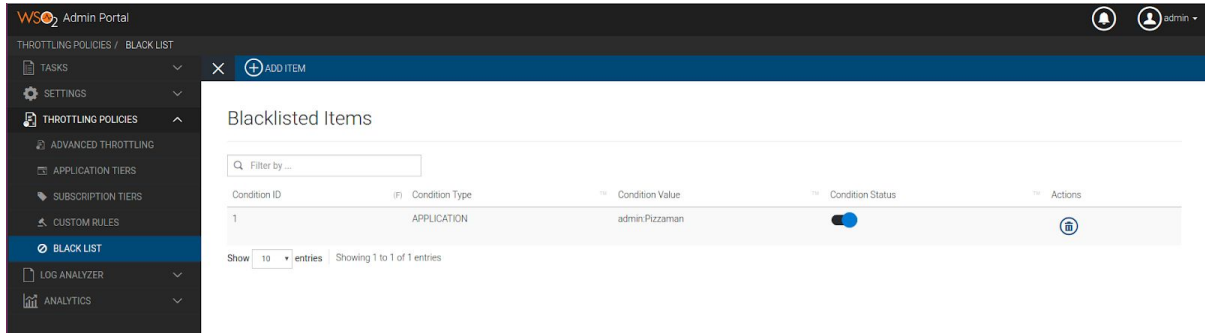
David:Pizzaman

Format :  
 \$(userName);\$(applicationName)  
 Eg : admin:DefaultApplication

Blacklist Cancel

Blacklisted Application will be listed as follows.





14. Now go to API Store again and invoke the **PizzaAPI** same as in step 11 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 PizzaAPI 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 2.2.0 to the same directory as the API Manager setup. **Note:** In order for statistics to be published, the `WSO2AM_STATS_DB` datasource in `<API-M_HOME>/repository/conf/datasources/master-datasources.xml` must be the same as the `WSO2AM_STATS_DB` datasource in `AM_ANALYTICS_HOME>/repository/conf/datasources/stats-datasources.xml`. Both of these datasources are the same by default.

3. Start the WSO2 APIM Analytics server, and then start the API Manager server.

For more information, see

<https://docs.wso2.com/display/AM220/Configuring+APIM+Analytics>

#### 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.

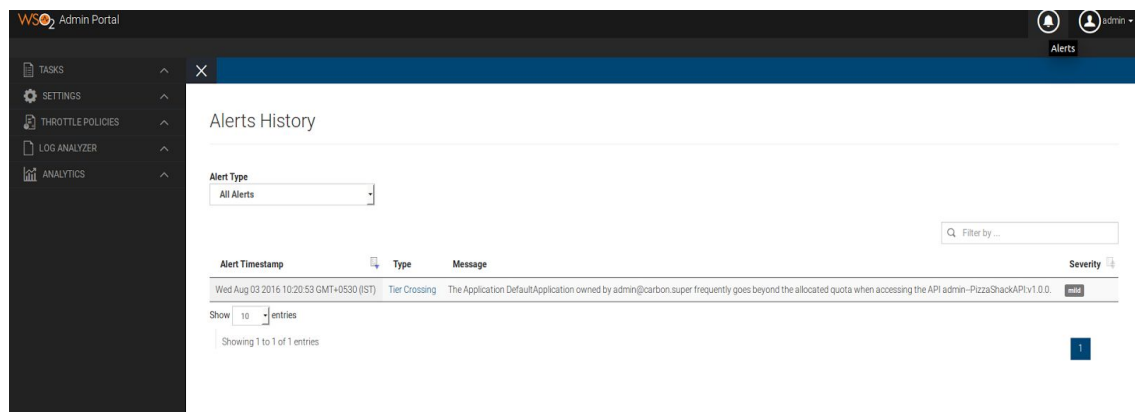
### High Level Steps

- Generate and view alerts
- Configure alert generation related parameters

### Generate Alerts

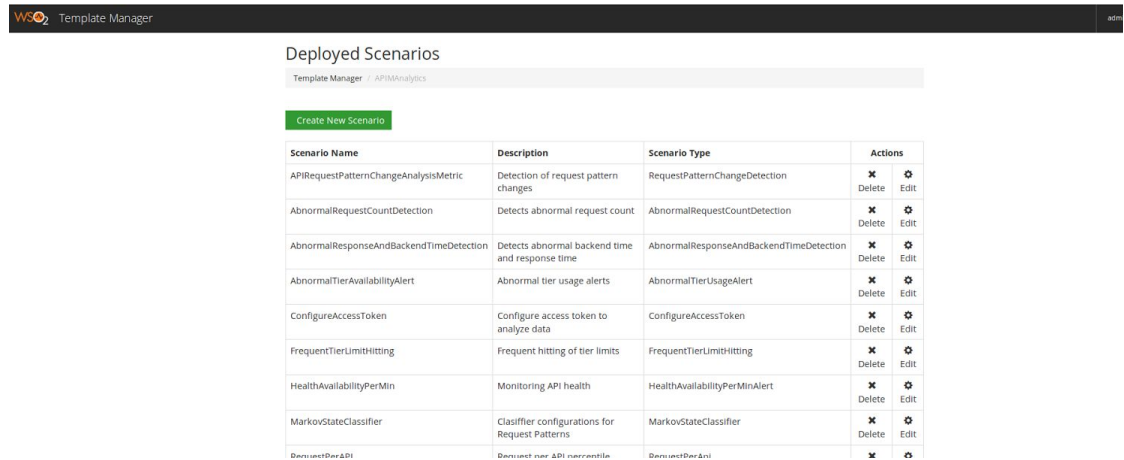
**Note:** API Manager and API Manager Analytics must be configured for analytics. This has been covered in the previous exercise.

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 PizzaShack API 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 PizzaShack API.
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. Log in 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

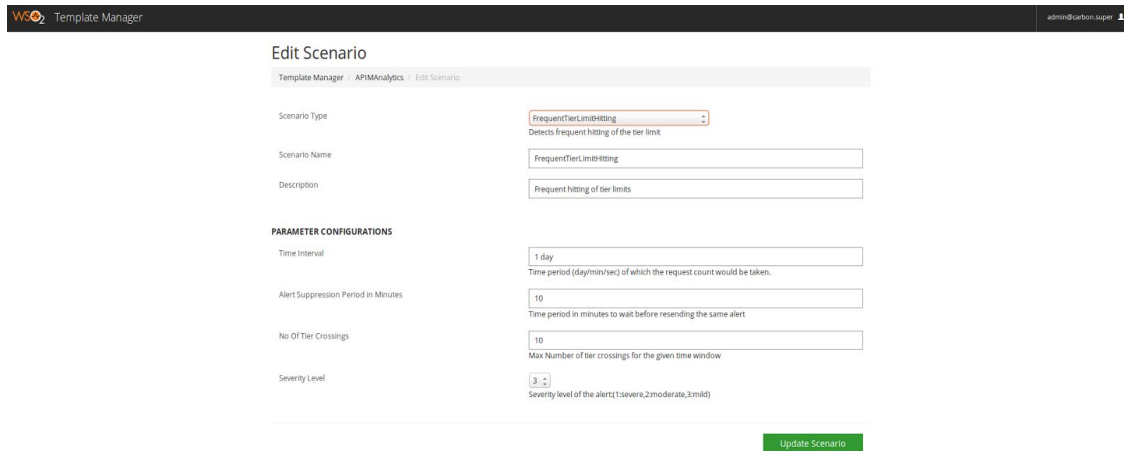
1. Log in to WSO2 API Manager Analytics carbon console (<https://localhost:9444/carbon>).
2. Select **Dashboard > Template Manager**.
3. In the **Template Manager**, select **APIMAnalytics**. This will open a configuration page for all the alert types.



The screenshot shows the 'Template Manager' interface with the 'APIMAnalytics' tab selected. A table titled 'Deployed Scenarios' lists various alert types with their descriptions and actions (Delete and Edit).

Scenario Name	Description	Scenario Type	Actions
APIRequestPatternChangeAnalysisMetric	Detection of request pattern changes	RequestPatternChangeDetection	Delete Edit
AbnormalRequestCountDetection	Detects abnormal request count	AbnormalRequestCountDetection	Delete Edit
AbnormalResponseAndBackendTimeDetection	Detects abnormal backend time and response time	AbnormalResponseAndBackendTimeDetection	Delete Edit
AbnormalTierAvailabilityAlert	Abnormal tier usage alerts	AbnormalTierUsageAlert	Delete Edit
ConfigureAccessToken	Configure access token to analyze data	ConfigureAccessToken	Delete Edit
FrequentTierLimitHitting	Frequent hitting of tier limits	FrequentTierLimitHitting	Delete Edit
HealthAvailabilityPerMin	Monitoring API health	HealthAvailabilityPerMinAlert	Delete Edit
MarkovStateClassifier	Classifier configurations for Request Patterns	MarkovStateClassifier	Delete Edit
RequestPerAPI	Request per API scenario	RequestPerAPI	Delete Edit

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



The screenshot shows the 'Edit Scenario' page for the 'FrequentTierLimitHitting' scenario. The page includes fields for Scenario Type, Scenario Name, and Description. Below these are the 'PARAMETER CONFIGURATIONS' for the scenario.

PARAMETER CONFIGURATIONS	
Time Interval	1 day Time period (day/min/sec) of which the request count would be taken.
Alert Suppression Period in Minutes	10 Time period in minutes to wait before resending the same alert
No. Of Tier Crossings	10 Max Number of tier crossings for the given time window
Severity Level	3 Severity level of the alert(1:severe,2:moderate,3:mild)

Update Scenario

## 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/v0.12/register
```

```
~ — java • wso2server.sh ... ~/Documents/wso2am-2.2.0/bin — -bash +
bin $ curl -k -X POST -H "Authorization: Basic YWRtaW46YWRtaW4=" -H "Content-Type: application/json" -
d @payload.json https://localhost:9443/client-registration/v0.12/register
```

The following response is displayed.

```
~ — java • wso2server.sh ... ~/Documents/wso2am-2.2.0/bin — -bash +
[bin $ curl -k -X POST -H "Authorization: Basic YWRtaW46YWRtaW4=" -H "Content-Type: application/json" -
d @payload.json https://localhost:9443/client-registration/v0.12/register
{"clientId":"vr0Wnpnq4nODPfmRhSDHrWPpbQQa","clientName":"admin_rest_api_store","callbackURL":"www.goo
gle.lk","clientSecret":"M5IUCWxokINKT7SANGfcx04SoPQa","isSaasApplication":true,"appOwner":null,"jsonS
tring":{"grant_types":"password refresh_token"}}]
bin $
```

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.

### Encode to Base64 format

Simply use the form below

vr0Wnpnq4nODPfmRhSDHrWPpbQQa:M5IUCWxokINKT7SANGfcx04SoPQa

> ENCODE <

UTF-8

You may also select output charset.

☐ Live mode OFF
 

Encodes while you type or paste (strict format).

Encodes an entire file (max. 10MB).

dnlwV25wbnE0bk9EUGZtUmhTREhyV1BwYFRYTpNNUiVQ1d4b2tJTktUN1NBbmdmY3gwNFNvUFFh

**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>" https://127.0.0.1:8243/token
```

```
~ — java • wso2server.sh ... ~/Documents/wso2am-2.2.0/bin — -bash +
[bin $curl -k -d "grant_type=password&username=admin&password=admin&scope=apim:api_create" -H "Authori
zation: Basic dnIwV25wbmE0bk9EUGZtUmhTREhyV1BwY1FRYTpNNU1VQ1d4b2tJTktUN1NBbmdmY3gwNFNvUFFh" https://1
27.0.0.1:8243/token
{"access_token":"3f11e0ef-dfff-367d-b613-56c16fd8af03","refresh_token":"473512e1-8536-320f-88dc-ce725
26ea74c","scope":"apim:api_create","token_type":"Bearer","expires_in":3584}bin $
```

**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"
}

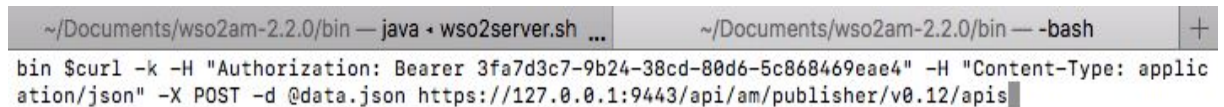
```

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/v0.12/apis

```



```

~/Documents/wso2am-2.2.0/bin — java • wso2server.sh ...  ~/Documents/wso2am-2.2.0/bin — -bash +
bin $curl -k -H "Authorization: Bearer 3fa7d3c7-9b24-38cd-80d6-5c868469eae4" -H "Content-Type: applic
ation/json" -X POST -d @data.json https://127.0.0.1:9443/api/am/publisher/v0.12/apis

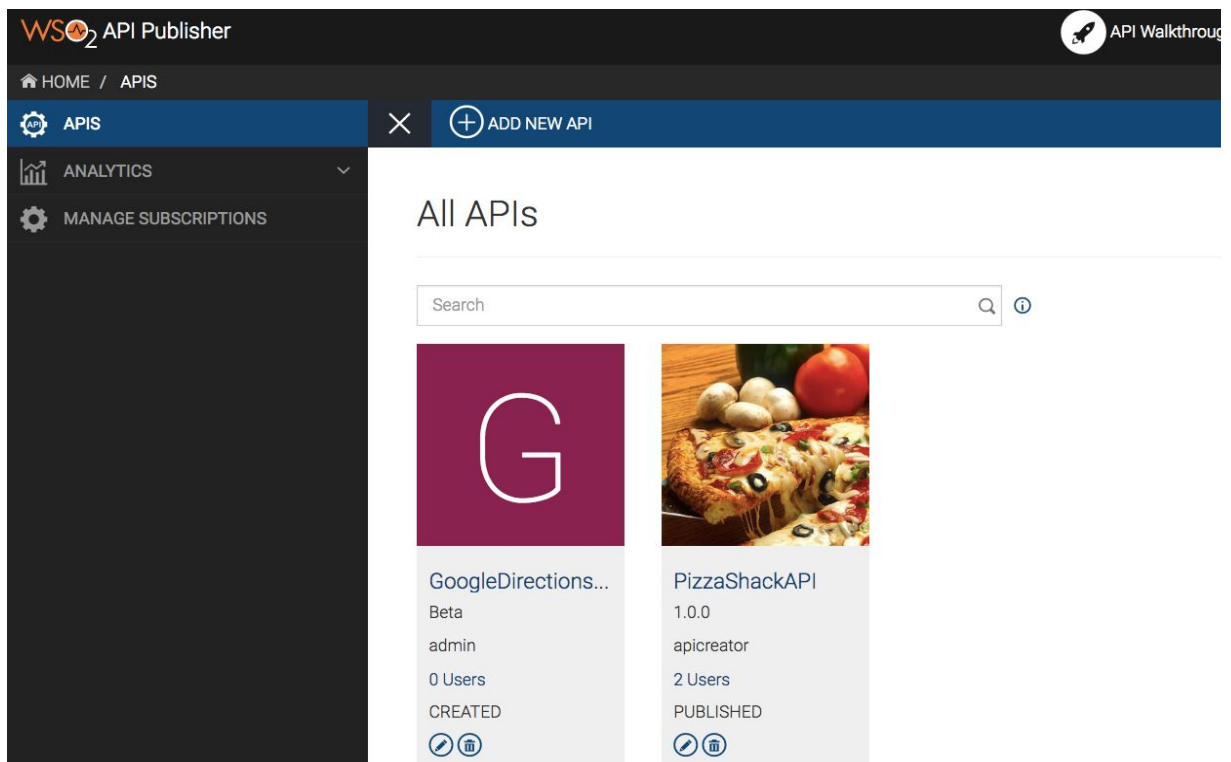
```



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

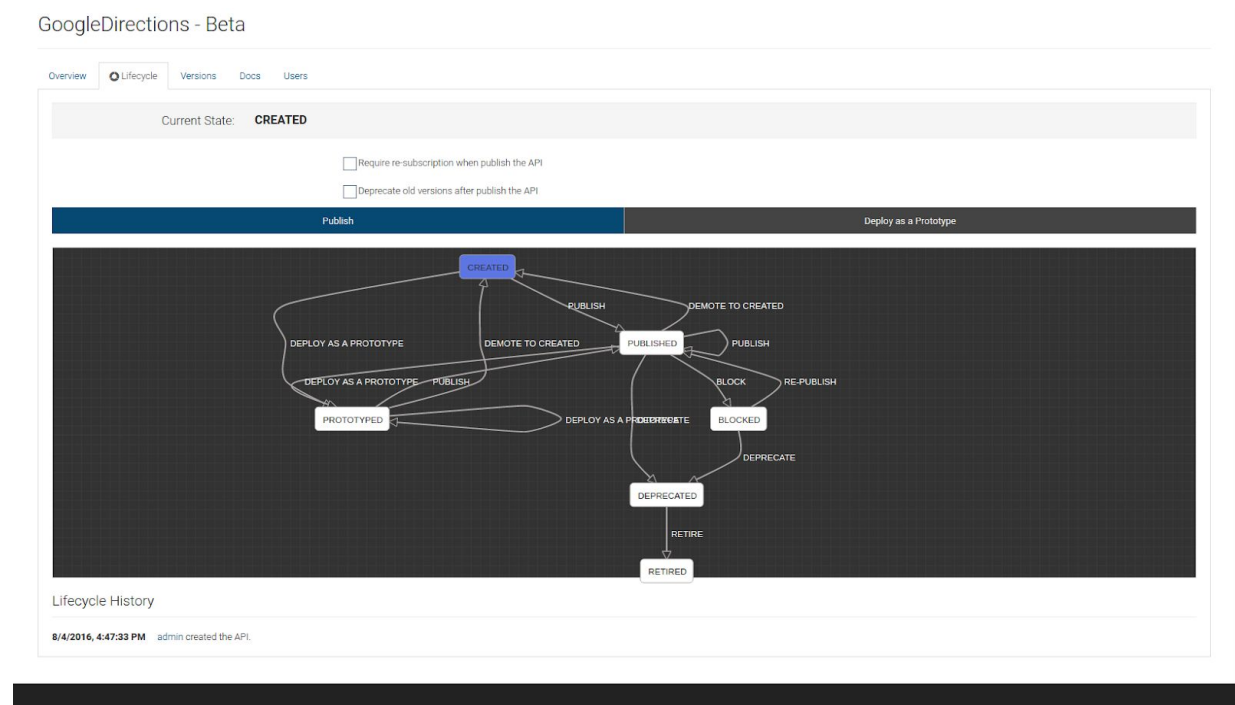
```
~/Documents/wso2am-2.2.0/bin — java • wso2server.sh ...  ~/Documents/wso2am-2.2.0/bin — -bash +
[bin $curl -k -H "Authorization: Bearer 3fa7d3c7-9b24-38cd-80d6-5c868469eae4" -H "Content-Type: applic
ation/json" -X POST -d @data.json https://127.0.0.1:9443/api/am/publisher/v0.12/apis
{"id":"240e5fd3-10d9-48e2-ac89-7c53166358f9","name":"GoogleDirectionsAPI","description":"Calculates d
irections between locations","context":"/googledirections","version":"Beta","provider":"admin","apiDe
finition":{"paths":{"/*":{"get":{"x-auth-type":"Application","x-throttling-tier":"Unlim
ited"},"responses":{"200":{"description":"OK"}}}}},"x-wso2-security":{"apim":{"x-wso2-sc
opes":[]},"swagger":"2.0","info":{"title":"GoogleDirectionsAPI","description":"Calcula
tes directions between locations"},"contact":{"email":"ApiPublisher@pizzashack.com","name":"
ApiPublisher"},"version":"Beta"},"wsdlUri":null,"status":"CREATED","responseCaching":"Disabled
","cacheTimeout":300,"destinationStatsEnabled":null,"isDefaultVersion":true,"type":"HTTP","transport"
:["http","https"],"tags":["multimedia","phone","mobile"],"tiers":["Bronze","Gold"],"apiLevelPolicy":n
ull,"maxTps":null,"thumbnailUri":null,"visibility":"PUBLIC","visibleRoles":[],"accessControl":"NONE",
"accessControlRoles":[],"visibleTenants":[],"endpointConfig":{"production_endpoints":{"url":"ht
tp://maps.google.com/maps/api/directions/"},"config":null},"endpoint_type":"http"},"endpointS
ecurity":null,"gatewayEnvironments":["Production and Sandbox"],"sequences":[],"subscriptionAvailability
":null,"subscriptionAvailableTenants":[],"businessInformation":{"technicalOwnerEmail":"ApiCreator@piz
zashack.com","businessOwnerEmail":"ApiPublisher@pizzashack.com","businessOwner":"ApiPublisher","techn
icalOwner":"ApiCreator"},"corsConfiguration":{"accessControlAllowOrigins":["*"],"accessControlAllowCr
edentials":false,"corsConfigurationEnabled":false,"accessControlAllowHeaders":["authorization","Acces
s-Control-Allow-Origin","Content-Type","SOAPAction"],"accessControlAllowMethods":["GET","PUT","POST",
"DELETE","PATCH","OPTIONS"]},"additionalProperties":{}}bin $
```

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_publish" -H "Authorization: Basic <encoded value>"
https://127.0.0.1:8243/token
```

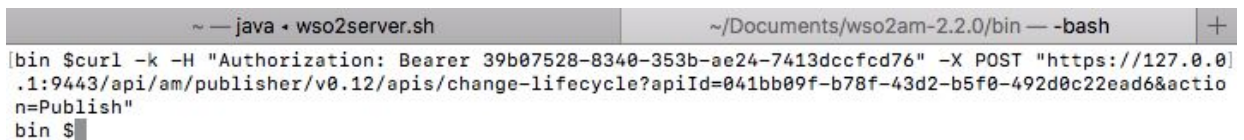
```

~ — java • wso2server.sh ... ~/Documents/wso2am-2.2.0/bin — -bash +
[bin $curl -k -d "grant_type=password&username=admin&password=admin&scope=apim:api_publish" -H "Authori
ization: Basic N05WSEwxMDZDeWdjckJWR2xTMV91dUZfa25FYTpzeU5VdW1UZ0ZDZTRNNHVxN1pMajUxWHlneTRh" https://
127.0.0.1:8243/token
{"access_token":"a93f9c07-84a2-3b54-933b-c967da59d377","refresh_token":"2a90e7c4-8d34-366d-9a0f-f01f3
db17dcb","scope":"apim:api_publish","token_type":"Bearer","expires_in":3600}bin $

```

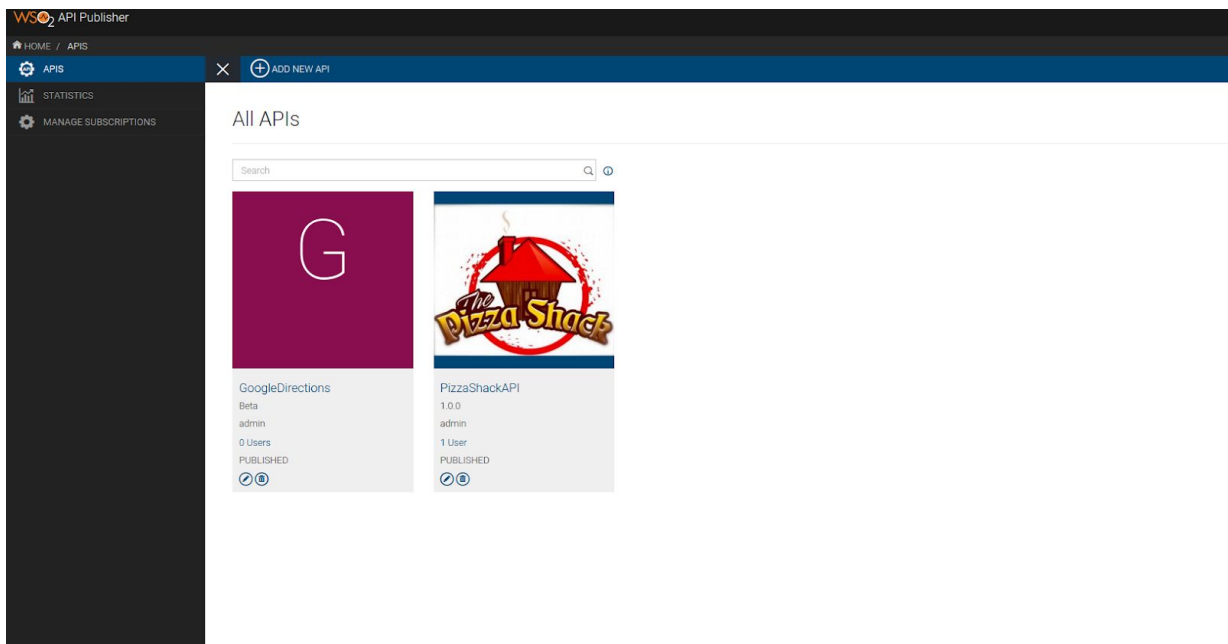
4. Type the following cURL command to publish the API. **Modify the Authorization Bearer token with the generated access token**, and **"apild"** with the id of the GoogleDirectionsAPI which can be found in the json response retrieved when the API is created.

```
curl -k -H "Authorization: Bearer aefcf3e9efb6bc09d34451a0824ed6e8" -X POST "https://127.0.0.1:9443/api/am/publisher/v0.12/apis/change-lifecycle?apiId=4d0b513e-71d4-489e-9681-31a9178bc189&action=Publish"
```



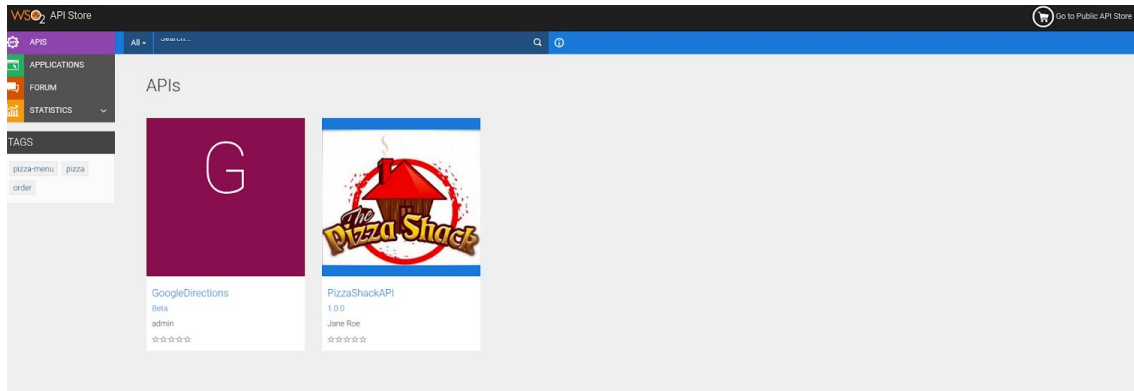
```
~ — java • wso2server.sh | ~/Documents/wso2am-2.2.0/bin — -bash | +
bin $ curl -k -H "Authorization: Bearer 39b07528-8340-353b-ae24-7413dcccfd76" -X POST "https://127.0.0.1:9443/api/am/publisher/v0.12/apis/change-lifecycle?apiId=041bb09f-b78f-43d2-b5f0-492d0c22ead6&action=Publish"
bin $
```

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/v0.12/register
```

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:sub
scribe" -H "Authorization: Basic
ZktWNWFJTxFkclFDRHduV1NMOExvbnRITm84YTpqUm9tUkdWUGhXMnVQZ0Fvd1Yzde
pBVzU5eThh" https://127.0.0.1:8243/token
```



```

~ — java • wso2server.sh ... ~/Documents/wso2am-2.2.0/bin — -bash
[bin $ curl -k -X POST -H "Authorization: Basic YWRtaW46YWRtaW4=" -H "Content-Type: application/json" -d @payload.json https://localhost:9443/client-registration/v0.12/register
[{"clientId": "CU12KMXJqEqEN0xHHR0Fth50Jica", "clientName": "admin_rest_api_store", "callBackURL": "www.google.lk", "clientSecret": "ZceeD3uTEwXC6zzQs7zRKJb11wwa", "isSaasApplication": true, "appOwner": "admin", "jsonString": "{\"grant_types\": \"password refresh_token\", \"redirect_uris\": \"www.google.lk\", \"client_name\": \"admin_rest_api_store\"}"}]
[bin $
[bin $
[bin $
[bin $ curl -k -d "grant_type=password&username=admin&password=admin&scope=apim:subscribe" -H "Authorization: Basic Q1UxMktNWExRXFFtJB4SEhSMEZ0aDVPSmljYTpY2V1RDN1VEV3WEM2enpRczd6UktKYjExd3dh" https://127.0.0.1:8243/token
{"access_token": "1a25930d-09f9-37e1-af43-54ade3f20d51", "refresh_token": "a9d75bf6-2414-3e05-9420-3589e0c5c1c1", "scope": "apim:subscribe", "token_type": "Bearer", "expires_in": 3600}
[bin $

```

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

```

{
  "groupId": "",
  "subscriber": "admin",
  "throttlingTier": "Unlimited",
  "description": "GoogleDirectionsAPI 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/v0.12/applications"

```

```

~ — java • wso2server.sh ... ~/Documents/wso2am-2.2.0/bin — -bash
[bin $ curl -k -H "Authorization: Bearer 1a25930d-09f9-37e1-af43-54ade3f20d51" -H "Content-Type: application/json" -X POST -d @data.json https://127.0.0.1:9443/api/am/store/v0.12/applications"
{"applicationId": "c1e21bc2-94ce-4db8-9993-9f11135bf8f8", "name": "GoogleDirectionsAPI", "subscriber": "admin", "throttlingTier": "Unlimited", "callbackUrl": null, "description": "GoogleDirectionsAPI App", "status": "APPROVED", "groupId": "", "keys": []}
[bin $

```

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

### Applications

An application is a logical collection of APIs. Applications allow you to use a single access token to invoke a collection of APIs and to subscribe to one API multiple times with different SLA levels. The DefaultApplication is pre-created and allows unlimited access by default.

Name	Tier	Workflow Status	Subscriptions	Actions
GoogleDirectionsAPI	Unlimited	ACTIVE	0	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
PizzaShack	Unlimited	ACTIVE	1	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
DefaultApplication	Unlimited	ACTIVE	0	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>

Show  entries Showing 1 to 3 of 3 entries

## 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
ZktWNWFJTxFkc1FDRHduV1NMOExvbnRITm84YTpqUm9tUkdWUGhXMnVQZ
0Fvd1YzdEpBVzU5eThh" https://127.0.0.1:8243/token
```

```
~ — java • wso2server.sh ... ~/Documents/wso2am-2.2.0/bin — -bash +
[bin $curl -k -d "grant_type=password&username=admin&password=admin&scope=apim:subscribe" -H "Authoriz
ation: Basic Q1UxMktNWExRXFFtjB4SEhSMZ0aDVPSmljYTpaY2VlRDNI1EV3WEM2enpRczd6UktKYjExd3dh" https://12
7.0.0.1:8243/token
{"access_token":"1a25930d-09f9-37e1-af43-54ade3f20d51","refresh_token":"a9d75bf6-2414-3e05-9420-3589e
0c5c1c1","scope":"apim:subscribe","token_type":"Bearer","expires_in":3342}bin $
```

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

```
curl -k -H "Authorization: Bearer
d1a167e580e40a8a2ae297fc4656677c"
"https://127.0.0.1:9443/api/am/store/v0.12/applications"
```

```
~ — java • wso2server.sh ... ~/Documents/wso2am-2.2.0/bin — -bash +
[bin $curl -k -H "Authorization: Bearer 1a25930d-09f9-37e1-af43-54ade3f20d51" "https://127.0.0.1:9443/
api/am/store/v0.12/applications"
{"count":2,"next":"","previous":"","list":[{"applicationId":"e3365531-6bdf-4472-88e0-f0c192f9f73d","n
ame":"DefaultApplication","subscriber":"admin","throttlingTier":"Unlimited","description":null,"statu
s":"APPROVED","groupId":""}, {"applicationId":"c1e21bc2-94ce-4db8-9993-9f11135bf8f8","name":"GoogleDir
ectionsAPI","subscriber":"admin","throttlingTier":"Unlimited","description":"GoogleDirectionsAPI App"
,"status":"APPROVED","groupId":""}]}bin $
```

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

```
curl -k -H "Authorization: Bearer
d1a167e580e40a8a2ae297fc4656677c"
https://127.0.0.1:9443/api/am/store/v0.12/apis
```

```
~ — java • wso2server.sh ... ~/Documents/wso2am-2.2.0/bin — -bash +
[bin $curl -k -H "Authorization: Bearer 1a25930d-09f9-37e1-af43-54ade3f20d51" https://127.0.0.1:9443/a
pi/am/store/v0.12/apis
{"count":2,"next":"","previous":"","list":[{"id":"041bb09f-b78f-43d2-b5f0-492d0c22ead6","name":"Googl
eDirectionsAPI","description":"Calculates directions between locations","context":"/googledirections/
Beta","version":"Beta","provider":"admin","status":"PUBLISHED","thumbnailUri":null,"scopes":[]},{id
:"24598003-9544-466c-b357-286a717b21c6","name":"PizzaShackAPI","description":"This is a RESTFul API f
or Pizza Shack online pizza delivery store.\r\n","context":"/pizzashack/1.0.0","version":"1.0.0","pro
vider":"apicreator","status":"PUBLISHED","thumbnailUri":"/apis/24598003-9544-466c-b357-286a717b21c6/t
humbnail","scopes":[{"key":null,"name":"order_pizza","roles":[]},{key":null,"name":"order_pizza","ro
les":[]}]}, {"key":null,"name":"order_pizza","roles":[]}], "pagination":{"total":2,"offset":0,"limit":25}}bin $
```

- Replace the text in the [API-M\_HOME]/bin/data.json file with the following. Give the retrieved apidIdentifier and applicationId.

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

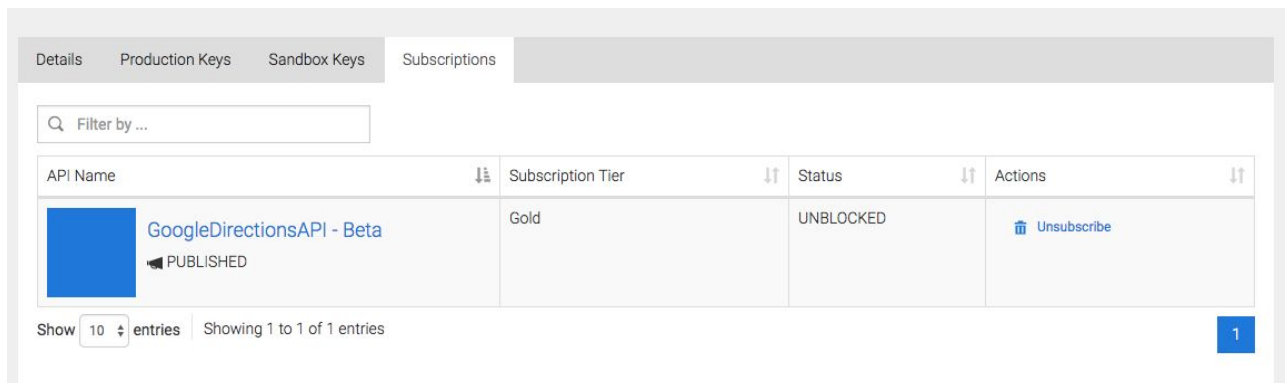
- Type the following cURL command to create a new subscription replacing the Authorization Bearer value.



```
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/v0.12/subscriptions"
```



```
~ — java • wso2server.sh ... ~/Documents/wso2am-2.2.0/bin — -bash
[bin $ curl -k -H "Authorization: Bearer 1a25930d-09f9-37e1-af43-54ade3f20d51" -H "Content-Type: applic
ation/json" -X POST -d @data.json "https://127.0.0.1:9443/api/am/store/v0.12/subscriptions"
{"subscriptionId": "0143e158-aa9a-4d39-bc44-ceb9c65dd2b3", "applicationId": "e3365531-6bdf-4472-88e0-f0c
192f9f73d", "apiIdentifier": "admin-GoogleDirectionsAPI-Beta", "tier": "Gold", "status": "UNBLOCKED"}bin $
```

- Go to the Store and click **APPLICATIONS**.
- Select the GoogleDirectionsAPI from the list. Select the **Subscriptions** tab. There will be a subscription tier field with a Gold Subscription.



Details   Production Keys   Sandbox Keys   Subscriptions				
Filter by ...				
API Name	Subscription Tier	Status	Actions	
 <b>GoogleDirectionsAPI - Beta</b> <small>PUBLISHED</small>	Gold	UNBLOCKED	 Unsubscribe	

Show 10 entries   Showing 1 to 1 of 1 entries

-END-