



## Workshop – SALA A

Construindo plataformas de Negócios  
usando o Marketplace de APIs.



---

Updated Nov, 2018



## Sumário

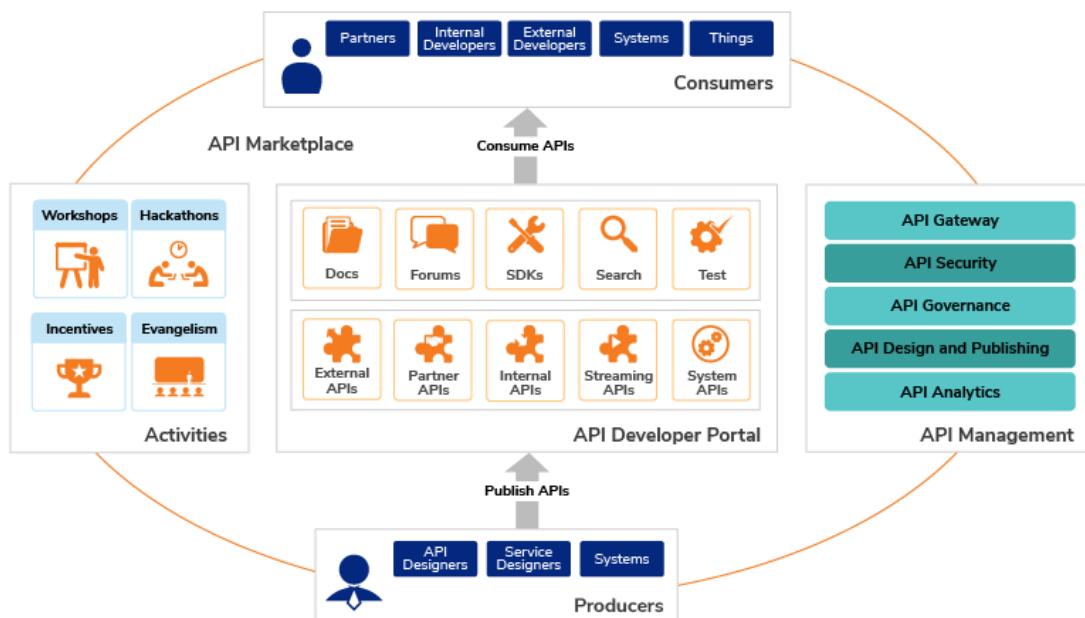
<b>APIM Introdução .....</b>	<b>2</b>
<b>Preparação do Ambiente .....</b>	<b>3</b>
<b>Demo 1 – API Quick-Start .....</b>	<b>8</b>
<b>Demo 2 – Map the Parameters of your Backend URLs with the API Publisher URLs .....</b>	<b>14</b>
<b>Demo 3 – Create a Prototyped API with an Inline Script.....</b>	<b>23</b>
<b>Demo 4 – Use the Community Features.....</b>	<b>27</b>
<b>Demo 5 – Adding Internationalization and Localization (para casa) .....</b>	<b>32</b>
<b>Demo 6 – Configuring API Monetization Category Labels.....</b>	<b>34</b>
<b>Demo 7 – Configure Multiple Tenants .....</b>	<b>36</b>
<b>Demo 7 – Customizing the API Store .....</b>	<b>36</b>
<b>Demo 8 – Migrating the APIs and Applications to a Different Environment .....</b>	<b>40</b>
<b>Demo 9 – Microgateway Quick Start.....</b>	<b>47</b>
<b>Demo 10 – Deploying the API Microgateway in Kubernetes.....</b>	<b>54</b>

# APIM Introdução

API Management é a tecnologia predominante para a maioria das organizações que desejam se tornar Digital Driven. Cada vez mais organizações utilizam o gerenciamento de API como um passo importante para se tornar digital.

Um API Marketplace, no entanto, vai além do simples gerenciamento de APIs; um mercado envolve a tecnologia, os negócios e os aspectos humanos do gerenciamento de API, garantindo que as APIs alcancem o objetivo a que se destinam: o consumo e o uso de APIs. Assim, o Marketplace de APIs permite a conexão entre produtores e consumidores de APIs e garante que essa conexão funcione.

Como parte deste workshop, vamos dar uma olhada no que significa API Management e como isso se relaciona com o conceito de um Marketplace corporativo. Analisaremos as vantagens da criação de um Marketplace de API corporativo e as etapas para o lançamento de um Marketplace eficaz de APIs corporativas. Para ilustrar ainda mais esse conceito, abordaremos os aspectos práticos do Marketplace de API da empresa Pizzashark.



# Preparação do Ambiente

## Instalação do JAVA

Instale [Oracle Java SE Development Kit \(JDK\)](#) 1.8.\* e configure a variável de ambiente JAVA\_HOME

```
$ /usr/libexec/java_home -V  
vi ~/.bash_profile  
# Settings for JAVA_HOME  
JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk1.8.0_191.jdk/Contents/Home  
export JAVA_HOME  
$ java -version  
java version "1.8.0_191"  
Java(TM) SE Runtime Environment (build 1.8.0_191-b12)  
Java HotSpot(TM) 64-Bit Server VM (build 25.191-b12, mixed mode)
```

## Instalação do API-M 2.6.0

Copie o pacote baixado no diretório /opt/wso2/summit2018

```
$ cd /summit2018  
$ unzip -x wso2am-2.6.0.zip
```



## Instalação do APIM Analytics

Copie o pacote baixado no diretório /opt/wso2/summit2018



```
$ cd /opt/wso2/summit2018  
$ unzip -x wso2am-analytics-2.6.0.zip
```

```
└── wso2am-analytics-2.6.0  
    ├── bin  
    ├── conf  
    ├── deployment  
    ├── lib  
    ├── resources  
    ├── tmp  
    ├── updates  
    └── wso2
```

### Instalação do API-M CLI

Copie o pacote baixado no diretório /opt/wso2/summit2018

```
$ tar xvf apimcli-1.2.0-macosx-x64.tar  
x apimcli/  
x apimcli/LICENSE  
x apimcli/README.html  
x apimcli/apimcli
```

### Instalação do API M Developer Studio e extras (Postman, VisualStudio, Curl)

Basta executar o pacote baixado.

Seu diretório deve estar assim.

```
/opt/wso2/summit2018/  
├── apimcli  
└── wso2am-2.6.0  
    ├── bin  
    ├── business-processes  
    ├── dbscripts  
    ├── lib  
    ├── modules  
    ├── repository  
    ├── resources  
    ├── samples  
    ├── tmp  
    ├── updates  
    └── wso2am-analytics-2.6.0  
        ├── bin  
        ├── conf  
        ├── deployment  
        ├── lib  
        ├── resources  
        ├── tmp  
        ├── updates  
        └── wso2
```



## Configurando o Analytics

```
$ cd /opt/wso2(summit2018/wso2am-2.6.0/repository/conf/datasources  
$ vi master-datasources.xml  
Procure por  
<datasource>  
    <name>WSO2AM_STATS_DB</name>  
E altere a linha URL.  
<configuration>  
    <url>jdbc:h2:$/opt/wso2(summit2018/wso2am-analytics-  
2.6.0/wso2/worker/database/WSO2AM_STATS_DB;AUTO_SERVER=TRUE</url>  
...  
...
```

Vá para o diretório /opt/wso2(summit2018/wso2am-2.6.0/repository/conf e edite o arquivo api-manager.xml

```
$ vi api-manager.xml  
Procure por Analytics e altere a tag Enabled.  
<Analytics>  
    <!-- Enable Analytics for API Manager -->  
    <Enabled>true</Enabled>
```

## Iniciando o API-M Analytics

```
$ cd /opt/wso2(summit2018/wso2am-analytics-2.6.0/bin  
$ worker.sh  
Output:
```



```
INFO {org.wso2.carbon.kernel.internal.CarbonStartupHandler} - WSO2 API Manager Analytics Server started in 5,461 sec
```

## Iniciando o API-M All-in-one profile

```
$ cd /opt/wso2/summit2018/wso2am-2.6.0/bin
```

```
$ ./wso2server.sh
```

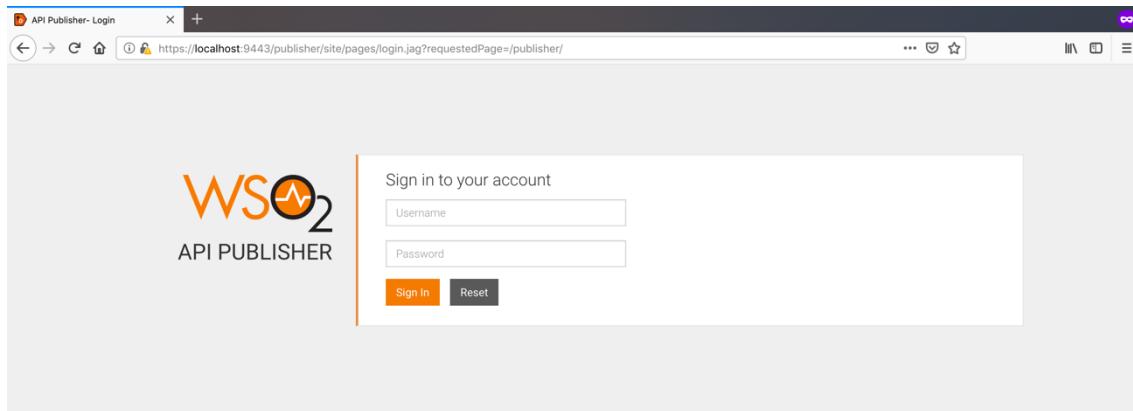
### Output:

```
...
INFO - StartupFinalizerServiceComponent Server : WSO2 API Manager-2.6.0
INFO - StartupFinalizerServiceComponent WSO2 Carbon started in 45 sec
INFO - CarbonUIServiceComponent Mgt Console URL : https://your.ip:9443/carbon/
INFO - CarbonUIServiceComponent API Store Default Context : https://your.ip:9443/store
INFO - CarbonUIServiceComponent API Publisher Default Context : https://your.ip:9443/publisher
...
```

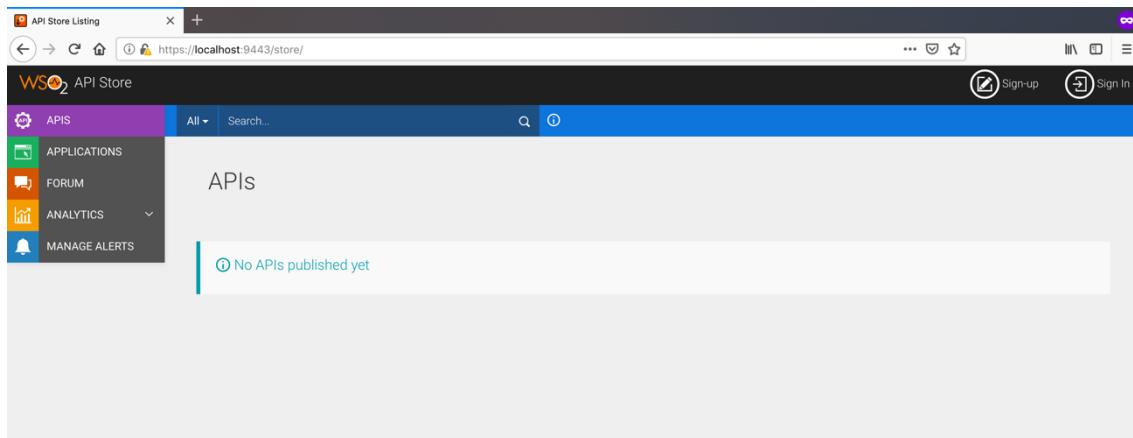
Neste momento você será capaz de acessar as consoles abaixo em seu browser:

Usuário e senha padrão: admin/admin

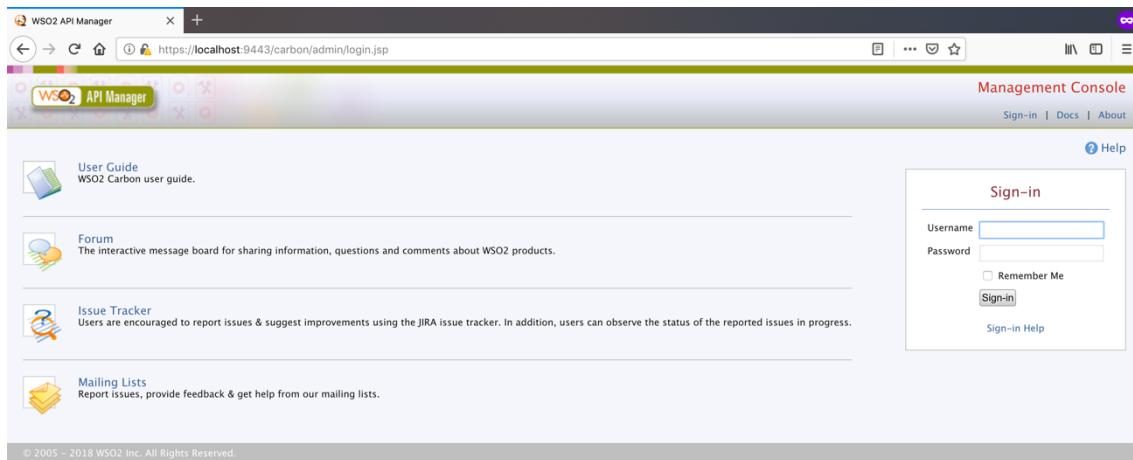
URL: <https://localhost:9443/publisher>



URL: <https://localhost:9443/store>



URL: <https://localhost:9443/carbon>



## Configurando o APIM Tooling:

Edite o arquivo /Applications/Eclipse.app/Contents/Info.plist

Procure pela chave <key>Eclipse</key>

Adicione a linha:

```
<string>-  
vm</string><string>/Library/Java/JavaVirtualMachines/jdk1.8.0_191.jdk/Contents/H  
ome/bin/java</string>
```



# Demo 1 – API Quick-Start

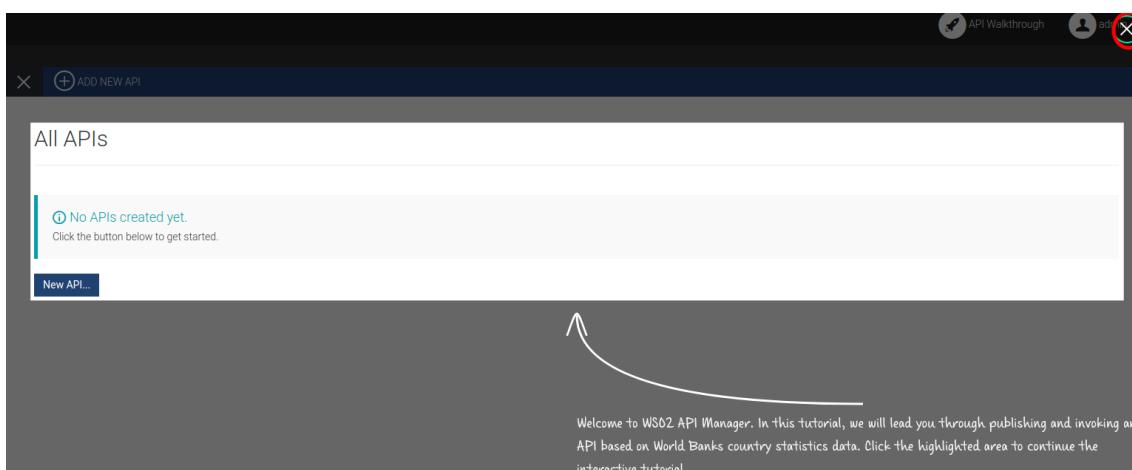
**API Publisher:** Enables API providers to publish APIs, share documentation, provision API keys and gather feedback on features, quality and usage. You access the Web interface via <https://<Server Host>:9443/publisher>.

Let's go through the use cases of the API Manager:

## Invoking your first API

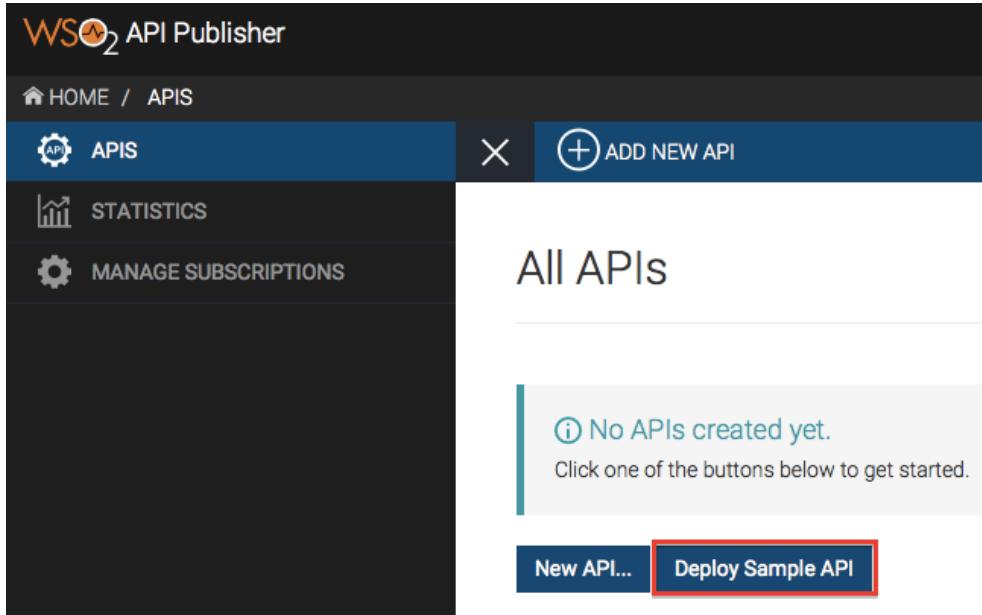
Follow the steps in this section to quickly deploy a sample API, publish it, subscribe to it, and invoke it.

1. Open the API Publisher (<https://<hostname>:9443/publisher>) and sign in with **admin/admin** credentials.
2. Exit from API creation tutorial by clicking the close icon(X) on top right corner.



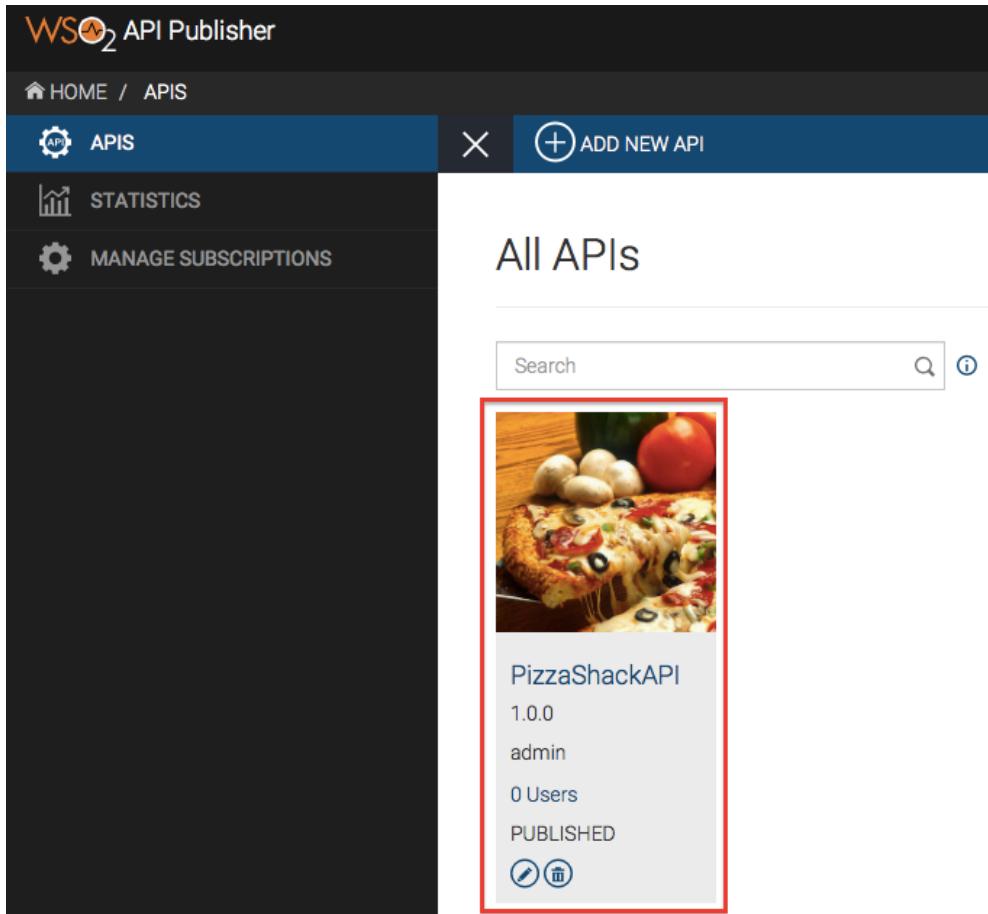
3. Click the **Deploy Sample API** button. It deploys a sample API called `PizzaShackAPI` into the API Manager.

This **Deploy Sample API** option is available only when there are no APIs in API Publisher. If you have already created a API, this option will not be available.



The screenshot shows the WSO2 API Publisher interface. The top navigation bar includes links for HOME, APIS, STATISTICS, and MANAGE SUBSCRIPTIONS. A prominent blue button on the right says '+ ADD NEW API'. The main content area is titled 'All APIs' and displays a message: 'No APIs created yet.' Below this message are two buttons: 'New API...' and 'Deploy Sample API', with 'Deploy Sample API' being highlighted with a red border.

4. Click `PizzaShackAPI` to open it.



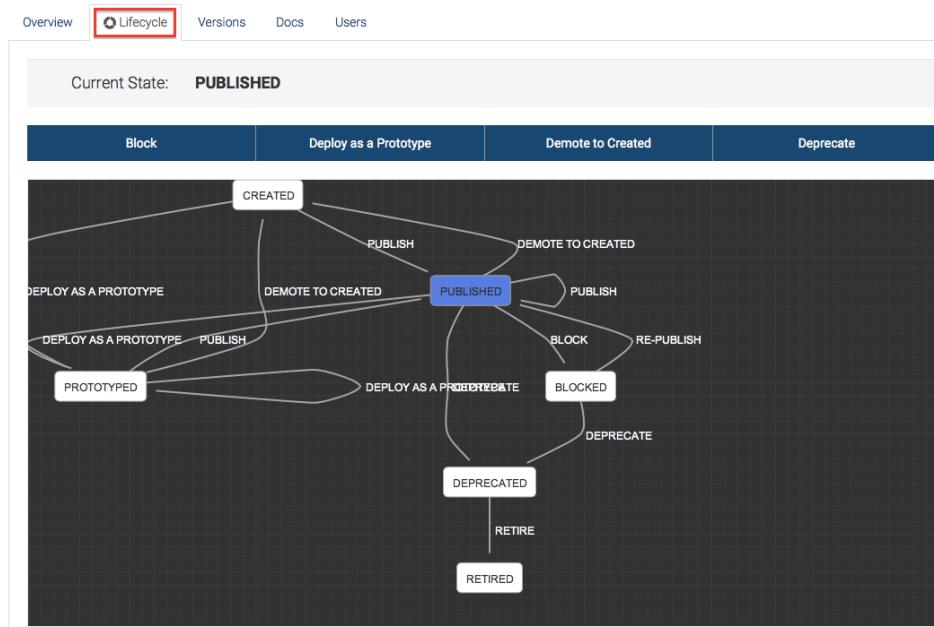
The screenshot shows the WSO2 API Publisher interface displaying the details for the 'PizzaShackAPI'. The API is shown as a card with the following information:

- Thumbnail image of a pizza.
- Name: `PizzaShackAPI`
- Version: 1.0.0
- Owner: admin
- Users: 0 Users
- Status: PUBLISHED
- Actions: Edit and Delete icons.

The entire card is highlighted with a red border.

5. Go to the **Lifecycle** tab and note that the **State** is `PUBLISHED`. The API is already published to the API Store.

## PizzaShackAPI - 1.0.0



6. Sign in to the API Store (<https://<hostname>:9443/store>) with the **admin/admin** credentials and click on the PizzaShackAPI API.

WSO<sub>2</sub> API Store

APIS APIs All ▾ Search...

APPLICATIONS FORUM STATISTICS

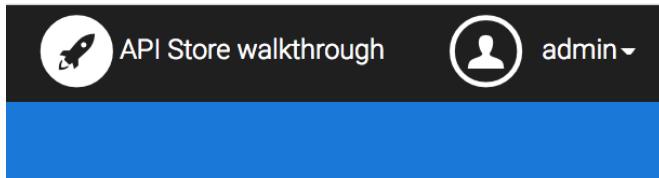
TAGS pizza

The screenshot shows the WSO2 API Store interface with the following details:

- APIs** section is selected.
- A search bar contains the text "pizza".
- A card for the "PizzaShackAPI" is displayed:
  - PizzaShackAPI**
  - 1.0.0**
  - Jane Roe**
  - ★★★★★**

## API Store Walkthrough

You can click "API Store walkthrough" to view the interactive tutorial to invoke the API.



7. Select the default application and an available tier, and click **Subscribe**.

A screenshot of the WSO2 API Store application details page for "PizzaShackAPI - 1.0.0". The page has a dark blue header with the WSO2 logo and "API Store". A sidebar on the left shows "APPLICATIONS" selected, along with "FORUM" and "STATISTICS". The main content area shows the API title and a thumbnail image of a pizza. To the right, there's a box with API metadata: Version 1.0.0, By Jane Roe, Updated 21/Jul/2016 13:41:55 PM IST, Status PUBLISHED, and Rating 5 stars. On the far right, there's a "Subscriptions" section with a dropdown for "Applications" set to "DefaultApplication" and a dropdown for "Tiers" set to "Unlimited". A red box highlights the "Subscribe" button.

8. When the subscription is successful, click **View Subscriptions** on the information message that appears. Click the **Production Keys** tab and click **Generate Keys** to generate an [access token](#) to invoke the API.

A screenshot of the WSO2 API Store "DefaultApplication" production keys generation page. The sidebar shows "APPLICATIONS" selected. The main content area has tabs: "Details" (disabled), "Production Keys" (highlighted with a red box), "Sandbox Keys", and "Subscriptions". A message box says "No Keys Found" with the subtext "No keys are generated for this type in this application." Below this, there's a "Grant Types" section with checkboxes for SAML2, Client Credential, IWA-NTLM, Code, Implicit, Password, and Refresh Token. Most checkboxes have a checked mark. A "Callback URL" input field is empty. An "Access token validity period" input field contains "3600" and a dropdown "Seconds.". A red box highlights the "Generate keys" button.

You have now successfully subscribed to an API. Let's invoke the API using the integrated Swagger-based API Console.

9. Click the APIs menu again and click the PizzaShackAPI to open it. When the API opens, click its API Console tab.

The screenshot shows the API console for the PizzaShackAPI version 1.0.0. The top right corner displays the API name and version. Below it, there's a summary card with details like Version: 1.0.0, By: Jane Roe, Updated: 21/Jul/2016 13:41:55 PM IST, Status: PUBLISHED, and Rating: ★★★★☆. To the right, there are dropdowns for Applications (Select Application...) and Tiers (Unlimited), and a 'Subscribe' button. At the bottom, tabs for Overview, API Console (which is highlighted with a red border), Documentation, and Forum are visible. Below these tabs, there are dropdowns for Try (DefaultApplication) and Using (Production). A Set Request Header field contains Authorization : Bearer e502e041-1455-358a-b8f8-57c413693b78.

Expand the GET method (which retrieves the menu) and click **Try it out**.

The screenshot shows the GET /menu endpoint. It includes implementation notes (Return a list of available menu items), a response class (Status 200 OK, List of APIs is returned), and a response model example:

```
{  
  "list": [  
    {  
      "price": "string",  
      "description": "string",  
      "name": "string",  
      "image": "string"  
    }  
  ]  
}
```

Below this, there's a 'Response Content Type' dropdown set to application/json. The 'Headers' section shows a table with columns for Header, Description, Type, and Other. The 'Response Messages' section lists 304 (Not Modified) and 406 (Not Acceptable, The requested media type is not supported). The 406 entry has a detailed description and a response model example:

```
{  
  "message": "string",  
  "error": [  
    {  
      "message": "string",  
      "code": 0  
    }  
  ],  
  "description": "string",  
  "code": 0,  
  "moreInfo": "string"  
}
```

At the bottom left, a red-bordered 'Try it out!' button is visible.

Note the response for the API invocation. It returns the list of menu items.

#### Response Body

```
[  
  {  
    "name": "BBQ Chicken Bacon",  
    "icon": "/images/6.png",  
    "description": "Grilled white chicken, hickory-smoked bacon and fresh sliced onions in barbecue sauce",  
    "price": "20.99"  
  },  
  {  
    "name": "Chicken Parmesan",  
    "icon": "/images/1.png",  
    "description": "Grilled chicken, fresh tomatoes, feta and mozzarella cheese",  
    "price": "20.99"  
  },  
  {  
    "name": "Chilly Chicken Cordon Bleu",  
    "icon": "/images/10.png",  
    "description": "Spinash Alfredo sauce topped with grilled chicken, ham, onions and mozzarella",  
    "price": "26.99"  
  },  
  {  
    "name": "Double Bacon &Cheese",  
    "icon": "/images/5.png",  
    "description": "Two strips of bacon, melted cheese and onions",  
    "price": "22.99"  
  }]
```

You have deployed a sample API, published it to the API Store, subscribed to it, and invoked the API using our integrated API Console.

# Demo 2 – Map the Parameters of your Backend URLs with the API Publisher URLs

This tutorial uses the [WSO2 API Manager Tooling Plug-in](#).

This tutorial explains how to map your backend URLs to the pattern that you want in the API Publisher. Note the following:

1. The URL pattern of the APIs in the Publisher  
is `http://<hostname>:8280/<context>/<version>/<API resource>`.
2. You can define variables as part of the URI template of your API's resources. For example, in the URI template `/business/{businessId}/address/`, `businessId` is a variable.
3. The variables in the resources are read during mediation runtime using property values with the "uri.var." prefix. For example, this HTTP endpoint gets the businessId that you specify in the resource  
`http://localhost:8280/businesses/{uri.var.businessId}/details`.
4. The URI template of the API's resource is automatically appended to the end of the HTTP endpoint at runtime. You can use the following mediator setting to remove the URL postfix from the backend endpoint: `<property name="REST_URL_POSTFIX" scope="axis2" action="remove"/>`.

We do the following mapping in this tutorial:



**Before you begin**, note that a mock backend implementation is set up in this tutorial for the purpose of demonstrating the API invocation. If you have a local API Manager setup, save [this file](#) in the `<APIM_HOME>/repository/deployment/server/synapse-configs/default/api` folder to set up the mock backend.

```
$ cp Response_API.xml /opt/wso2(summit2018/wso2am-2.6.0/repository/deployment/server/synapse-configs/default/api
```

Output from APIM Server:

```
INFO - API Initializing API: Response_API
INFO - DependencyTracker API : Response_API was added to the Synapse configuration successfully
INFO - APIDeployer API named 'Response_API' has been deployed from file :
/opt/wso2(summit2018/wso2am-2.6.0/repository/deployment/server/synapse-configs/default/api/Response_API.xml
```

1. Log in to the API Publisher, design a new API with the following information, click **Add** and then click **Next: Implement >**.

Field		Sample value
Name		TestAPI
Context		/test
Version		1.0.0
Visibility		Public
Resources	URL pattern	/business/{businessId}/address/
	Request types	GET

The screenshot shows the 'Design' tab of the WSO2 API Publisher interface. The 'General Details' section contains the following fields:

- Name: TestAPI
- Context: /test
- Version: 1.0.0
- Visibility: Public
- Description: (empty)
- Tags: (empty)

The 'Thumbnail Image' section shows a placeholder icon and a 'Select image' button. The 'Dimensions (max): 100 x 100 pixels' note is visible.

The 'API Definition' section includes:

- URL Pattern: /test/1.0.0 /business/{businessId}/address/
- Method selection:  GET,  POST,  PUT,  DELETE,  PATCH,  HEAD, more
- Add button: (+ Add)

At the bottom, there are 'Save' and 'Next: Implement >' buttons.

2.



3. The **Implement** tab opens. Give the information in the table below.

Field	Sample value
Endpoint type	HTTP/REST endpoint
Production endpoint	http://localhost:8280/businesses/{uri.var.businessId}/details
Sandbox endpoint	http://localhost:8280/businesses/{uri.var.businessId}/details

TestAPI: /test/1.0.0

The screenshot shows the WSO2 API Manager interface with the 'Implement' tab selected. The top navigation bar has three tabs: 'Design' (blue), 'Implement' (white), and 'Manage' (dark blue). Below the tabs, there's a section titled 'Managed API' with a sub-section 'Provide the production and sandbox endpoints of the API to be managed.' Under 'Endpoint Type', 'HTTP/REST Endpoint' is selected. There are checkboxes for 'Load Balanced' and 'Failover'. Two endpoint fields are shown: 'Production Endpoint' with the value 'http://localhost:8280/businesses/{uri.var.businessId}/details' and 'Sandbox Endpoint' with the value 'http://localhost:8280/businesses/{uri.var.businessId}/details'. Both endpoint fields have a 'Test' button next to them. A 'Show More Options' link is also present. Below this, there's a 'Message Mediation Policies' section with a checkbox 'Enable Message Mediation' and a note 'Check to select a message mediation policy to be executed in the message flow'. At the bottom left, there are 'Save' and 'Next : Manage >' buttons.

- 4.

5. Click **Next: Manage >** to go to the Manage tab, select the **Gold** tier and publish the API.

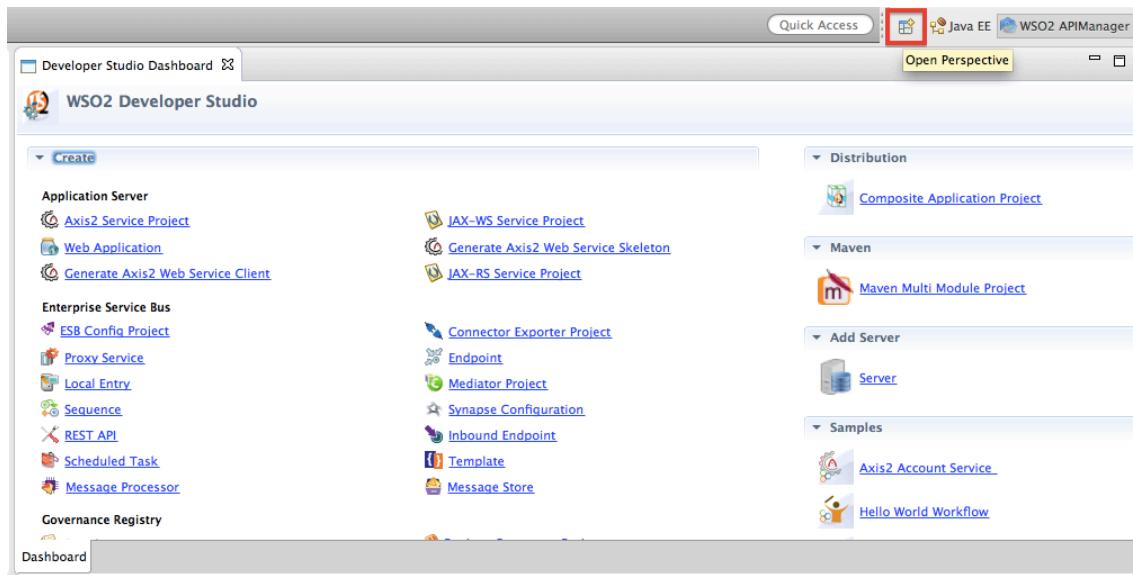
The screenshot shows the WSO2 API Manager configuration interface. At the top, there are three tabs: 'Design' (selected), 'Implement', and 'Manage'. Below the tabs, the 'Configurations' section contains settings for transports and response caching. In the 'Throttling Settings' section, it shows subscription tiers (Gold selected, Silver and Bronze available) and advanced throttling policies (Apply per Resource selected).

As the API's resource is appended to its endpoint by Synapse at runtime, let's write a custom sequence to remove this appended resource.

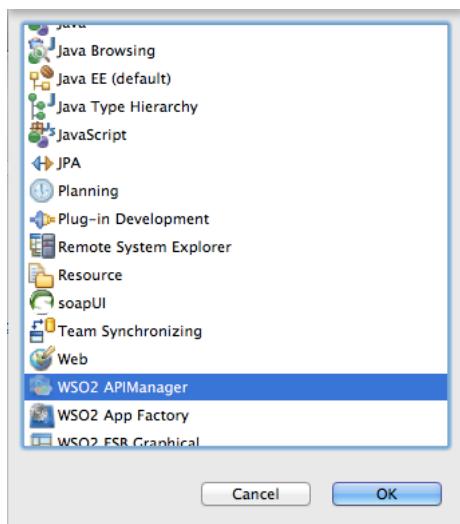
6. Copy the the following to a text editor and save the file in XML format (e.g., TestSequence.xml).

```
<sequence xmlns="http://ws.apache.org/ns/synapse" name="TestSequence">
    <property name="REST_URL_POSTFIX" scope="axis2" action="remove"/>
</sequence>
```

7. Download and install the [WSO2 API Manager Tooling Plug-in](#) if you have not done so already. Open Eclipse by double clicking the Eclipse.app file inside the downloaded folder.
8. Click **Window > Open Perspective > Other** to open the Eclipse perspective selection window. Alternatively, click the **Open Perspective** icon shown below at the top right corner.



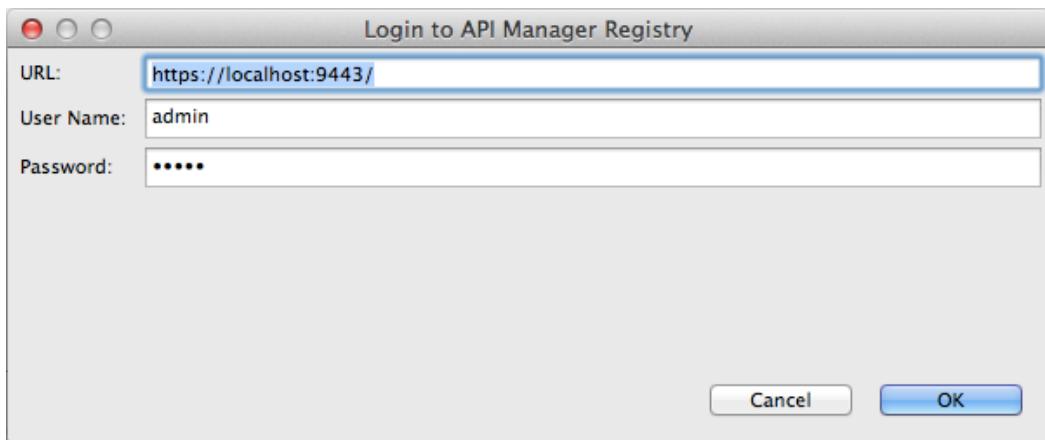
9. On the dialog box that appears, click **WSO2 APIManager** and click **OK**.



10. On the APIM perspective, click the **Login** icon as shown below.

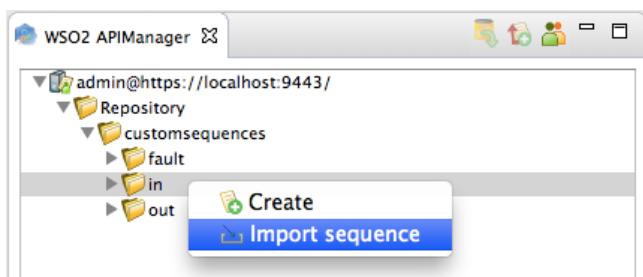


11. On the dialog box that appears, enter the URL, username and password of the Publisher server.



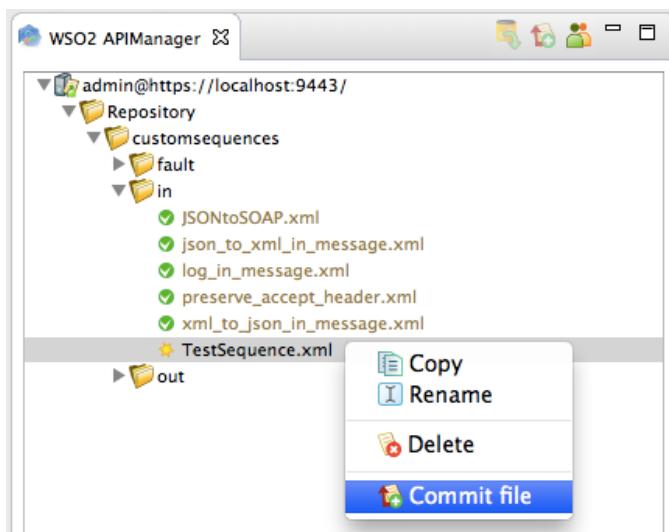
12. On the tree view that appears, expand the folder structure of the existing API.

13. Right-click on the `in` sequence folder and click **Import Sequence** to import the sequence you create above.



14. Browse to the `TestSequence.xml` file you created in step 4.

15. Your sequence now appears on the APIM perspective. Right-click on the imported sequence and click **Commit File** to push the changes to the Publisher server.



16. Log back into the API Publisher, click **Edit** and go to the **Implement** tab. Select the **Enable Message Mediation** check box and engage the **In** sequence that you created earlier.

The screenshot shows the 'Implement' tab of the WSO2 API Publisher. At the top, there are three tabs: 'Design' (selected), 'Implement' (highlighted in blue), and 'Manage'. Below the tabs, the 'Managed API' section is shown. Under 'Endpoint Type', 'HTTP/REST Endpoint' is selected. There are options for 'Load Balanced' and 'Failover'. The 'Production Endpoint' is set to 'http://localhost:8280/businesses/{uri.var.businessId}/details' and has a 'Test' button. The 'Sandbox Endpoint' is also set to the same URL and has a 'Test' button. A 'Show More Options' link is present. The 'Message Mediation Policies' section contains a checkbox for 'Enable Message Mediation' which is checked, and a dropdown menu showing 'In Flow' selected with 'TestSequence' chosen. Buttons for 'Upload In Flow', 'Upload Out Flow', and 'Upload Fault Flow' are available. The 'CORS configuration' section has an unchecked checkbox for 'Enable API based CORS Configuration'. At the bottom, there are 'Save' and 'Next : Manage >' buttons.

TestSequence.xml removes the URL postfix from the backend endpoint, since the URI template of the API's resource is automatically appended to the end of the URL at runtime. Therefore the **request** URL is modified by adding this sequence to the **In flow**.

17. Save and Publish the API.

You have created an API. Let's subscribe to the API and invoke it.

18. Log in to the API Store and subscribe to the API.

The screenshot shows the WSO2 API Store interface. It displays the details for 'TestAPI - 1.0.0'. On the left, there is a large purple square icon with a white 'T'. To its right, the API details are listed: Version 1.0.0, By admin, Updated 30/Jul/2016 11:59:27 AM IST, Status PUBLISHED, and Rating 5 stars. Below this, there are tabs for 'Overview', 'API Console', 'Documentation', and 'Forum'. On the right, there is a 'Subscribe' button. A red box highlights the 'DefaultApplication' dropdown under 'Applications' and the 'Gold' dropdown under 'Tiers'.

19. Click the **View Subscriptions** button when prompted. The **Subscriptions** tab opens.



20. Click the **Production Keys** tab and click **Generate Keys** to create an application access token. If you have already generated a token before, click **Regenerate** to renew the access token.

DefaultApplication

Details   **Production Keys**   Sandbox Keys   Subscriptions

Show Keys

Consumer Key  
.....

Consumer Secret  
.....

Grant Types

Application can use the following grant types to generate Access Tokens. Based on the application requirement, you can select one or more grant types.

SAML2    IWA-NTLM  
 Client Credential    Code

Callback URL  
.....

Update

Generating Access Tokens

The following cURL command shows how to generate an access token using the password grant type.

```
curl -k -d "grant_type=password&username=Username&password=Password" \
-H "Authorization: Basic Base64(consumer-key:consumer-secret)" \
https://192.168.1.6:8243/token
```

In a similar manner, you can generate an access token using the client credential grant type with the following cURL command.

```
curl -k -d "grant_type=client_credentials" \
-H "Authorization: Basic Base64(consumer-key:consumer-secret)" \
https://192.168.1.6:8243/token
```

Generate a Test Access Token

Access Token  
.....

Above token has a validity period of **3600** seconds. And the token has (**am\_application\_scope default**) scopes.

Scopes  
No Scopes Found..

Validity period  
3600   Seconds.

**Re-generate**

21. Click the API Console tab of your API.

TestAPI - 1.0.0

Version: 1.0.0  
By: admin  
Updated: 30/Jul/2016 11:59:27 AM IST  
Status: PUBLISHED  
Rating: ★★★★☆

Applications: Select Application...  
Tiers: Gold

Subscribe

Overview API Console Documentation Forum

Try: DefaultApplication  
Using: Production Key

Set Request Header: Authorization : Bearer a884b32e-764a-38f9-9b0e-824e59a5c8a0

Swagger ( /swagger.json )  
Show/Hide | List Operations | Expand Operations

default

GET /business/{businessId}/address/\*

22. Note that the businessId is added in the UI as a parameter. Give a businessId and click Try it out to invoke the API.

GET /business/{businessId}/address/\*

Parameters

Parameter	Value	Description	Parameter Type	Data Type
businessId	123		path	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
200			

Try it out!

23. Note the response that you get. According to the mock backend used in this tutorial, you get the response Received Request.

Response Body

<response><value>Received Request</value></response>

In this tutorial, you mapped the URL pattern of the APIs in the Publisher with the endpoint URL pattern of a sample backend.

# Demo 3 – Create a Prototyped API with an Inline Script

In this tutorial, you create a prototyped API with an inline script, deploy it as a prototype, and invoke it using the API Console integrated in the API Store. Typically, you create APIs with inline scripts for testing purposes. An API prototype is created for the purpose of early promotion and testing. You can deploy a new API or a new version of an existing API as a prototype. It gives subscribers an early implementation of the API that they can try out without a subscription or monetization, and provide feedback to improve. After a period of time, publishers can make changes that the users request and publish the API.

1. Sign in to the API Publisher using `admin` as the username and password.  
`https://<hostname>:9443/publisher` (e.g., `https://localhost:9443/publisher`).
2. Select the option to design a new REST API and click **Start Creating**.

Let's get started!

Add New API



The screenshot shows the 'Add New API' interface. There are four options: 'I Have an Existing API', 'I Have a SOAP Endpoint', 'Design a New REST API' (which is selected), and 'Design a New Websocket API'. Below these options is a 'Start Creating' button, which is highlighted with a red border. A 'Show More Options' link is also visible.

3. Give the information in the table below. To add resources, click the **Add** button. Since the URL Pattern used here is a variable, it is denoted within curly braces.

Field		Sample value
Name		Location_API
Context		/location
Version		1.0.0
Resources	URL pattern	{town}
	Request types	GET

## Design API

The screenshot shows the 'Design' tab selected in the top navigation bar. The 'General Details' section contains fields for Name (Location\_API), Context (/location), Version (1.0.0), Access Control (All), Visibility on Store (Public), Description (Maximum 20000 characters), and Tags (Add tags). A 'Thumbnail Image' section with a placeholder image and a 'Select image' button is also present. The 'API Definition' section shows a URL Pattern of /location/1.0.0/{town} with checkboxes for GET, POST, PUT, DELETE, PATCH, and HEAD methods, all of which are checked except for GET. A red box highlights the 'Add' button below the URL pattern.

4.

- After the resource is added, expand its **GET** method and note that a parameter by the name `town` is added under the resource. You use it to pass the payload to the backend. Once done, click **Next: Implement >**.

To specify multiple parameters in the API resource, separate the parameters with a forward slash.

The screenshot shows the expanded GET method for the URL pattern /location/1.0.0/{town}. The 'Summary' section is highlighted with a red box. Below the method list, there are 'Save' and 'Next: Implement >' buttons.

6. In the **Prototyped API** section under the **Implement** tab, select the implementation method as **Inline**.

Location\_API: /location/1.0.0

The screenshot shows the WSO2 API Manager interface. At the top, there are three tabs: 'Design' (blue), 'Implement' (white), and 'Manage' (light blue). The 'Implement' tab is active. Below the tabs, there are two sections: 'Managed API' and 'Prototyped API'. The 'Prototyped API' section is expanded, showing a note about using the inbuilt JavaScript engine to prototype the API. It includes a 'Implementation Method' dropdown with 'Inline' selected (highlighted by a red box) and 'Endpoint' as options. Below this, there's a 'Resources' section with a 'GET' method endpoint '/{town}' and a '+Summary' link. Further down, there's a 'CORS configuration' section with a checkbox for 'Enable API based CORS Configuration'. At the bottom right of the interface, there are 'Save' and 'Deploy as a Prototype' buttons.

The inline JavaScript engine does not provide support for SOAP APIs. If you opt for the endpoint implementation method instead of inline, you need to provide an endpoint to a prototype API. For example, <http://ws.cdyne.com/phoneverify/phoneverify.asmx>

7. Expand the `GET` method and give the following as the script. It reads the payload that the user sends with the API request and returns it as a JSON value. The value `mc` is the message context.

```
mc.setProperty('CONTENT_TYPE', 'application/json');
var town = mc.getProperty('uri.var.town');
mc.setPayloadJSON('{ "Town" : "'+town+'"');
```

The screenshot shows the expanded details for the GET method. It includes a 'Description' field with a link to '+Add Implementation Notes', a 'Response Content Type' field (set to 'application/json'), and a 'Parameters' table. The table has one row with 'Parameter Name' 'town', 'Description' '+Empty', 'Parameter Type' 'path', 'Data Type' 'string', and 'Required' 'True'. Below the table is a 'Script' section containing the provided JavaScript code. The code is numbered from 1 to 4.

Parameter Name	Description	Parameter Type	Data Type	Required
town	+Empty	path	string	True

```

1 mc.setProperty('CONTENT_TYPE', 'application/json');
2 var town = mc.getProperty('uri.var.town');
3 mc.setPayloadJSON('{ "Town" : "'+town+'"'});
4

```

8.

9. Click **Deploy as a Prototype**.

10. When prompted, choose to open the newly published API in the API Store.

**Tip:** You can invoke prototyped APIs without signing in to the API Store or subscribing to the API. The purpose of a prototype is advertising and giving an early implementation for users to test.

11. The Location API opens. Click the **API Console** tab.

The screenshot shows the 'Location\_API - 1.0.0' page. At the top right, there is a large red square icon with a white 'L' shape inside. To its right, the version information is displayed: Version: 1.0.0, By: admin, Updated: 27/Jul/2018 01:39:15 AM IST, Status: PROTOTYPED, and Rating: ★★★★☆. Below this, there are tabs for Overview, API Console (which is highlighted with a red border), Documentation, SDKs, and Forum.

12. Expand the GET method, click **Try it out**. Give any value for the town (e.g. London) and click **Execute** to invoke the API.

The screenshot shows the 'Try it out' dialog for the GET /{town} method. It has a 'Parameters' section where a parameter named 'town' is set to 'London'. Below this is a large blue button labeled 'Execute' which is highlighted with a red border. At the bottom, there are sections for 'Responses' (Code: 200) and 'Response content type' (application/json).

13. Note the payload you gave as a JSON output in the response.

The screenshot shows the response details for a 200 status code. It includes a 'Response body' section containing the JSON payload: { "Town": "London" }. Below this is a 'Response headers' section showing the header content-type: application/json; charset=UTF-8.

You have created an API with inline script, deployed it as a prototype and invoked it through the integrated API Console.

An API can also be prototyped by moving the API to the prototyped state in the API lifecycle. For more information, see the [Deploy and Test as a Prototype](#) tutorial.

# Demo 4 – Use the Community Features

The API Store provides several useful features to build and nurture an active community of users, for your APIs. This is required to advertise APIs, learn user requirements and market trends.

Let's see what community features are available in the API Store:

## Use the search facility

You can search for APIs in the API Publisher or Store in the following ways:

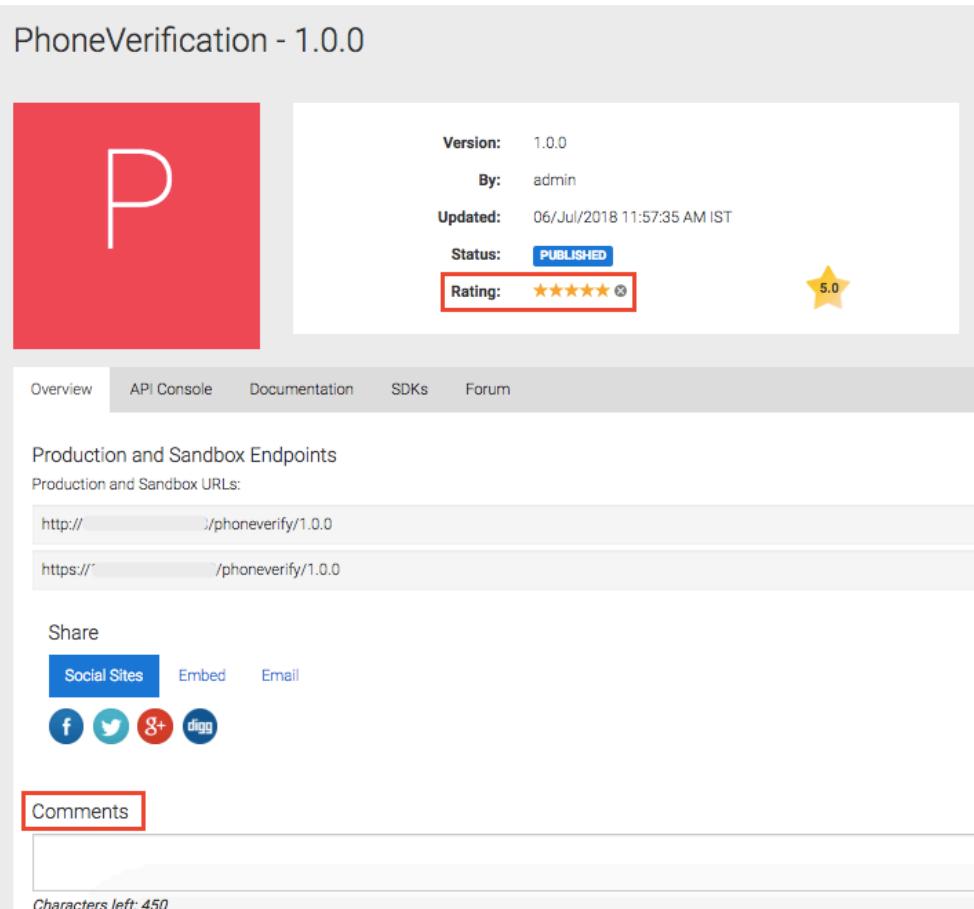
Clause	Syntax
By the API's name	As this is the default option, simply enter the API's name and search.
By the API provider	<b>provider:xxxx</b> . For example, provider:admin Provider is the user who created the API.
By the API version	<b>version:xxxx</b> . For example, version:1.0.0 A version is given to an API at the time it is created.
By the context	<b>context:xxxx</b> . For example, <a href="#">context:/phonverify</a> Context is the URL context of the API that is specified as /<context_name> at the time the API is created.
By the API's status	<b>status:xxxx</b> . For example, status: PUBLISHED A state is any stage of an API's lifecycle. The default lifecycle stages include created, prototyped, published, deprecated, retired and blocked.
By description	<b>description:xxxx</b> A description can be given to an API at the time it is created or later. There can be APIs without descriptions as this parameter is optional.
By the subcontext	<b>subcontext:xxxx</b> . For example, <a href="#">subcontext:/checkphonenumbers</a> . A subcontext is the URL pattern of any resource of the API. API resources are created at the time the API is created or later when it is modified. For example, if you create a resource by the name checkphonenumbers,

Clause	Syntax
	then /checkphonenumbers becomes one subcontext of the API.
By the content of the API documentation	<b>doc:xxxx</b> You can create API documentation in-line (using the API Publisher UI itself), by uploading a file or referring to an external URL. This search enables you to give a sentence or word phrase that is inside the in-line documentation and find the API that the documentation is added for.

### Rate and comment

Rates and comments give useful insights to potential API consumers on the quality and usefulness of an API. You can rate and comment on each API version.

1. Log in to the API Store and click on a published API.
2. The API's **Overview** page opens. Note the rating and commenting options there:



The screenshot shows the 'PhoneVerification - 1.0.0' API overview page. At the top right, there is a summary box with the following details:

- Version:** 1.0.0
- By:** admin
- Updated:** 06/Jul/2018 11:57:35 AM IST
- Status:** PUBLISHED
- Rating:** ★★★★★ 5.0

A yellow star icon is displayed next to the rating. Below this summary, there is a navigation bar with links: Overview, API Console, Documentation, SDKs, and Forum. The 'Overview' link is currently active. Under the navigation bar, there is a section for 'Production and Sandbox Endpoints' and 'Production and Sandbox URLs'. It lists two URLs: <http://>/phoneverify/1.0.0 and <https://>/phoneverify/1.0.0. Below these URLs, there is a 'Share' section with links for Social Sites, Embed, and Email, followed by social media icons for Facebook, Twitter, Google+, and Digg. At the bottom of the page, there is a red-bordered 'Comments' section with a text input field and a note indicating 'Characters left: 450'.

3. Add a rating and a comment. Note that the comments appear sorted by the time they were entered, alongside the author's name.

Comments

---

Characters left: 450

[Add](#)

comment1

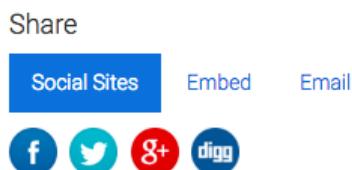
Posted By [admin](#) on July 30, 2016 11:13:58 AM GMT+05:30

comment2

Posted By [admin](#) on July 30, 2016 11:14:05 AM GMT+05:30

## Share on social media/e-mail

1. Log in to the API Store and click on a published API.
2. On the API's **Overview** page, you get the social media options using which you can share and advertise APIs.



## Embed an API widget

A widget is an embeddable version of the API in HTML that you can share on your website or other web pages. This is similar to how YouTube videos are embedded in a web page.

1. Log in to the API Store and click on a published API.

2. Note the **Embed** tab under the API's sharing options.

The screenshot shows the 'Overview' tab selected in the top navigation bar. Below it, there's a section for 'Production and Sandbox Endpoints' with two URLs listed: <http://10.100.7.109:8280/phonereverify/1.0.0> and <https://10.100.7.109:8243/phonereverify/1.0.0>. Under the 'Share' section, the 'Embed' tab is highlighted with a red box around its content. It contains an iframe code snippet:

```
<iframe width="450" height="120" src="https://localhost:9443/store/apis/widget?name=PhoneVerification&version=1.0.0&provider=admin&tenant=carbon.super" frameborder="0" allowfullscreen></iframe>
```

## Participate in the forum

1. Log in to the API Store.
2. Click the **Forum** tab to go to the forum, where you can initiate conversations and share your opinions with other users.

The screenshot shows the 'PhoneVerification - 1.0.0' API page. The 'Forum' tab is selected in the top navigation bar, highlighted with a red box. Below the tabs, there's a search bar and a 'Create New Topic' button. The main content area displays a message: 'No topics found. There are no forum topics available.' On the right side, there are sections for 'Applications' (DefaultApplication) and 'Tiers' (Unlimited), each with a dropdown menu. A 'Subscribe' button is also present.

Click **Create New Topic** to start a new topic in the Forum.

3. Enter the Subject and Description of your topic and click Create.

Create New Topic

Subject

New Forum Topic

Description

Description

A horizontal toolbar for rich text editing, featuring icons for bold (B), italic (I), underline (U), strikethrough (del), font color (A), font size (T), alignment (list), and various other document formats (table, image, video, etc.).

**Create** **Cancel**

# Demo 5 – Adding Internationalization and Localization (para casa)

The API Manager comes with two Web interfaces as API Publisher and API Store. The following steps show an example of how to localize the API Publisher UI. Same instructions apply to localize the API Store.

## Changing the browser settings

1. Follow the instructions in your Web browser's user guide and set the browser's language to a preferred one. For example, in Google Chrome, you set the language using the **Settings -> Show advanced settings -> Languages** menu.
2. Set the browser's encoding type to UTF-8.

## Introduction to resource files

1. Go to <APIM\_HOME>/repository/deployment/server/jaggeryapps/publisher directory where <APIM\_HOME> is the API Manager distribution's home.
2. There are two types of resource files used to define localization strings in the API Manager.
3. The resource file used to store the strings defined in .jag files according to browser locale (For example, locale\_en.json) is located in .../publisher/site/conf/locales/jaggery folder.
4. The resource file i18nResources.json, which is used to store strings defined in client-side javascript files such as pop-up messages when a UI event is triggered, is located in .../publisher/site/conf/locales/js folder.

For example,

```
./locales/jaggery:  
locale_en.json  
  
./locales/js:  
i18nResources.json
```

To implement localization support for jaggery, we use its in-built script module 'i18n'. For more information, refer to <http://jaggeryjs.org/documentation.jag?api=i18n>.

## Localizing strings in Jaggery files

To localize the API publisher to Spanish, first localize the strings defined in jaggery files. Create a new file by the name **locale\_{toLocaleCode}.json** inside



...publisher/site/conf/locales/jaggery folder. For example, if the language set in the browser is Spanish, the locale code is **es** and the file name should be **locale\_es.json**.

The file name which includes the locale code will differ according to the language selected in your browser. Create a new file for the language you select, if the selected language is not available. The file Name should be **locale\_{localeCode}.json** where the `localeCode` refers to the sub tag of the particular language. Please refer the [IANA Language Subtag Registry page](#) for a list of sub tags.

By matching the accept Language header, the particular file will be selected from the resource file directory of each web application for the localization by the server. If there is no locale file found with the matching sub tag, `locale_default.json` file will be served for the localization.

Add the key-value pairs to `locale_es.json` file. For an example on adding key value pairs, refer to `locale_en.json` file in ...publisher/site/conf/locales/jaggery folder. It is the default resource file for jaggery.

In addition, a section of a sample `locale_es.json` file is shown below for your reference.

```
{  
    "name" : "Nombre" ,  
    "context" : "Contexto" ,  
    "version" : "Versión" ,  
    "description" : "Descripción" ,  
    "visibility" : "Visibilidad" ,  
    "thumbnail" : "Uña del pulgar" ,  
    "endpoint" : "Producción URL" ,  
    "sandbox" : "Cajón de arena URL" ,  
    ..
```

#### Localizing strings in client-side Javascript files

1. To localize the javascript UI messages, navigate to publisher/site/conf/locales/js folder and update **i18nResources.json** file with relevant values for the key strings.
2. Once done, open the API Publisher web application in your browser (<https://<YourHostName>:9443/publisher>) .
3. Note that the UI is now changed to Spanish.

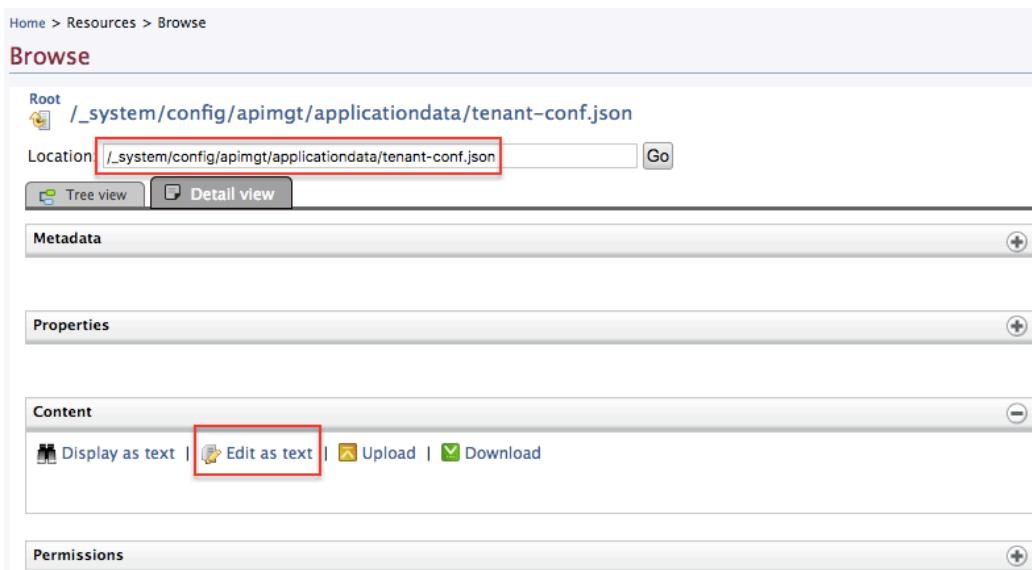
# Demo 6 – Configuring API Monetization Category Labels

When defining throttling tiers using the Admin Portal, you have the option to specify a given billing plan for tiers. A tier is defined as either a free or paid tier. Depending on the tiers available for a given API, the following API monetization categories are displayed as labels in the store.

- **Free** - If all subscription tiers are defined as Free, the API uses the **Free billing plan** and the API is labeled as Free in the Store.
- **Paid** - If all subscription tiers are defined as Paid, the API uses the **Commercial billing plan** and the API is labeled as Paid in the Store.
- **Freemium** - If the API has a combination of Free and Paid subscription tiers, the API uses the **Freemium billing plan** and the API is labeled as Freemium in the Store.

Follow the configuration steps below to enable API monetization category labels:

1. Sign in to the API Manager's Management Console (<https://<hostname>:9443/carbon>).
2. Navigate to the **Main** menu, and click **Browse**, which is under the **Resources** tab.
3. Enter the following path in the **Location:** text-box and click **Go**.  
`/_system/config/apimgt/applicationdata/tenant-conf.json`



The screenshot shows the 'Browse' interface of the WSO2 API Manager. The URL bar at the top shows 'Home > Resources > Browse'. The main area is titled 'Browse' with a 'Root' node. Below it, there is a 'Location' input field containing the path '/\_system/config/apimgt/applicationdata/tenant-conf.json', which is highlighted with a red box. There are two tabs: 'Tree view' and 'Detail view', with 'Detail view' selected. Below the tabs are two expandable sections: 'Metadata' and 'Properties'. At the bottom, there is a 'Content' panel with four buttons: 'Display as text', 'Edit as text' (which is also highlighted with a red box), 'Upload', and 'Download'. Below the Content panel is a 'Permissions' section.

4. In the **Contents** panel, click the **Edit as text** link and the `tenant-conf.json` file opens.
5. To enable monetization categories for APIs, set the `EnableMonetization` property to true. By default, it is set to false.

6. Define the subscription tiers as required.

For example if you are working with the unlimited tier,

- To define the unlimited tier as **paid**, set the `IsUnlimitedTierPaid` property to true.
- To define the unlimited tier as **free**, set the `IsUnlimitedTierPaid` property to false.

As Freemium APIs has a combination of paid and free subscription tiers, the configuration involved in defining the subscription tiers will be the same as above. However, Freemium APIs need to have a minimum of one subscription defined as paid and free.

7. After the edits, click **Save Content**.

Note that the above configuration can be done independently on a per tenant basis.

When the above `EnableMonetization` property is set to true for the respective tenant, the API monetization category labels are displayed in the tenant API store.

The screenshot shows the WSO2 API Store interface. The top navigation bar includes the WSO2 logo, a search bar, and a user icon. The left sidebar has tabs for APIS (selected), APPLICATIONS, FORUM, and STATISTICS, with a dropdown menu for TAGS containing the option 'pizza'. The main content area is titled 'APIs' and displays three API entries:

- Facebook**: Version 1.0.0, Admin, FREE. It features a large blue Facebook logo icon.
- PizzaShackAPI**: Version 1.0.0, Jane Roe, PAID. It features an image of a pizza with toppings like mushrooms and olives.
- WSO2Service**: Version 1.0.0, Admin, FREEMIUM. It features a circular orange icon with a black '2' and a white line graph.

Each API entry includes its name, version, owner, and a five-star rating.

#### Related links

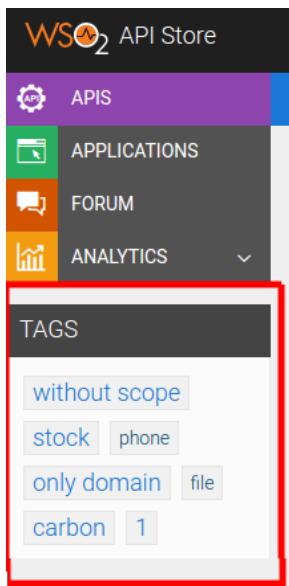
For more information on Monetization of APIs, see [Enabling Monetization of APIs-](#).

## ~~Demo 7 – Configure Multiple Tenants~~

### Demo 7 – Customizing the API Store

Categorizing and grouping APIs

API providers add tags to APIs when designing them using the API Publisher. Tags allow API providers to categorize APIs that have similar attributes. When a tagged API gets published to the API Store, its tags appear as clickable links to the API consumers, who can use them to quickly jump to a category of interest. The font size of the tag in the Store varies based on the number of APIs that are assigned to it. Therefore, for example the font size of a tag which has 10 APIs assigned to it will be bigger than the font size of a tag that has only 2 APIs assigned to it.



If you want to see the APIs grouped according to different topics in the API Store, add an API group:

Although the way in which you add a Tag and API group appears to be similar there are differences. Therefore, you need to note the following:

- The **group name should always have the suffix -group** and it **can have spaces** in it (e.g., APIs groups-group).
- The **tag name should not have a suffix or prefix**, but it **can have spaces**.

1. Go to <API-M\_HOME>/repository/deployment/server/jaggeryapps/store/site/conf directory, open the site.json file and set the tagWiseMode attribute as true.
2. Add an API group to the APIs that you wish to group.
  - a. Go to the API Publisher (<https://<HostName>:9443/publisher>).
  - b. Click on the edit link of the respective API as shown below.

The screenshot shows the WSO2 API Publisher interface. The top navigation bar includes 'HOME / APIS' (with 'APIS' highlighted), 'ANALYTICS', and 'MANAGE SUBSCRIPTIONS'. Below this is a search bar with a magnifying glass icon and a help icon. The main area is titled 'All APIs' and lists three items:

- Integration**: Version 1.0.0, created by admin, 0 users, PUBLISHED. Edit icon is highlighted with a red box.
- PizzaShackAPI**: Version 1.0.0, created by admin, 0 users, PUBLISHED. Edit icon is highlighted with a red box.
- Workflow**: Version 1.0.0, created by admin, 0 users, PUBLISHED. Edit icon is highlighted with a red box.

- c. Add a group name to the APIs that you wish to group.  
For example add the "APIs groups-group" tag to the Workflow and Integration APIs.
- d. Save the API for the tag to appear in the Store.
- e. Repeat steps 2 (a) to (d) to add another APIs to the newly created group.  
Sign in to the API Store and note the API groups.

The screenshot shows the WSO2 API Store interface. On the left, there's a sidebar with tabs for APIS, APPLICATIONS, FORUM, and ANALYTICS. Below that is a section for TAGS with categories like pizza, Workflow, Social Media, Integration, and Finance. The main area is titled "APIs groups" and shows three items: "Finance APIs", "Integrator", and "New APIs", each represented by a blue cloud icon with gears.

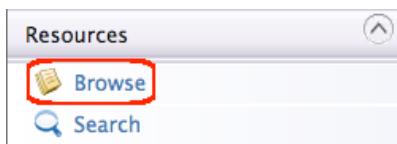
If you wish, you can click on a group to see the APIs that belong to a specific group.

This screenshot shows the "APIs - New APIs" page. At the top, it says "APIS GROUPS / NEW APIs". Below that, there are two API entries: "Integration" and "Workflow". Each entry has a thumbnail image (a vertical bar for Integration and a stylized 'W' for Workflow), a version number (1.0.0), a creator name (admin), and a five-star rating.

#### Customizing the API group (optional)

If you want to change the descriptions and the thumbnail images that come by default, do the following:

1. Sign in to the Management Console and click the **Resources > Browse** menu to open the registry.



2. Create a collection named `tags` under the registry location `/_system/governance/apimgt/applicationdata`.



Root [/\\_system/governance/apimgt/applicationdata](/_system/governance/apimgt/applicationdata)

Location: /\_system/governance/apimgt/applicationdata

**Metadata**

**Properties**

**Entries**

Add Resource  
 Add Collection (highlighted with a red box)  
 Create Link

**Add Collection**

Name *	<input type="text" value="tags"/>
Media Type	<input type="button" value="not specified"/>
Description	<input type="text"/>

3. Give read permission to the system/wso2.anonymous.role role.

**Entries**

Add Resource  
 Add Collection (highlighted with a red box)  
 Create Link

**Permissions**

**New Role Permissions**

Role	<input type="button" value="system/wso2.anonymous.role"/>	Action	<input type="button" value="Read"/>	<input checked="" type="radio"/> Allow <input type="radio"/> Deny
<input type="button" value="Add Permission"/>				

4. Add each tag as collections under the tags collection (e.g., Workflow APIs-group, Integration APIs-group, Quote APIs-group.)
5. Navigate to each tag collection and upload the following:
  - **description.txt** with the description of the tag
  - **thumbnail.png** for the thumbnail image
6. Back in the API Store, note the changes you did in the registry.

# Demo 8 – Migrating the APIs and Applications to a Different Environment

WSO2 API Manager (WSO2 API-M) allows you to maintain multiple environments running on the same WSO2 API-M version. This feature allows you to import and export applications or APIs between your environments. For example, if you have an application or API running in the development environment, you can import it and export to the test(QA) environment. Thereby, APIs and applications do not have to be created from scratch in different environments.

## Getting Started

After running the CLI tool make sure to add an environment before you start working with the import/export CLI commands, because all APIs and applications need to be imported or exported to a specific environment.

## Running the CLI tool

### Step 1 - Deploy the API import/export tool

1. Download the latest version of WSO2 API Manager from <http://wso2.com/products/api-manager/>.
2. Start WSO2 API Manager.
3. Download the latest WSO2 API import/export tool (`api-import-export-2.6.0-v1.war`) from [here](#).
4. Note that the import/export tool attached is specific to this version of WSO2 API Manager.
5. Copy the downloaded `api-import-export-2.6.0-v1.war` file to the `<API-M_HOME>/repository/deployment/server/webapps` folder.

The file is automatically deployed as hot deployment is enabled.

### Step 2- Run the CLI tool

1. Navigate to the API Management Tooling page - <https://wso2.com/api-management/tooling/>



2. Click **Download** under **CLI**.
3. Select a generated archive suitable for your platform (i.e., Mac, Windows, Linux) and extract it to the CLI tool that you downloaded to a desired location and `cd` into it.
4. Navigate to the working directory where the executable CLI Tool resides.
5. Execute the following command to start the CLI tool.
6. `./apimcli`
7. Add the location of the extracted folder to your system's `$PATH` variable to be able to access the executable from anywhere.
8. For further instructions execute the following command.

```
apimcli -help
```

Global flags for CLI tool

The following are some global flags that you can use with the CLI tool.

`--verbose`

Enable verbose logs (Provides more information on execution)

`--insecure, -k`

Allow connections to SSL sites without certs

`--help, -h`

Display information and example usage of a command

#### Adding an environment

You can add environments by either manually editing the `$HOME/.wso2apimcli/main_config.yaml` file or by running the following CLI command.

```
$ apimcli add-env
```

The directory structure for the configuration files (`$HOME/.wso2apimcli`) will be created upon execution of the `apimcli` command.

1. Make sure that WSO2 API Manager is started and that the CLI import/export tool is running.
2. Run the following CLI command to add an environment.

```
apimcli add-env -n <environment-name> \
                --registration <registration-endpoint> \
                --apim <API-Manager-endpoint> \
                --token <token-endpoint> \
                --import-export <endpoint-for-environment> \
                --admin <admin-REST-API-endpoint> \
                --api_list <API-listing-REST-API-endpoint> \
                --app_list <application-listing-REST-API-
                endpoint>
```

### 3. Flags:

Required flags:

--name, -n

There are no short flags for the following flags.

```
--registration
--apim
--token
--import-export
--admin
--api_list
--app_list
```

Removing an environment

1. Make sure that WSO2 API Manager is started and that the CLI import/export tool is running.
2. Run the following CLI command to remove an environment.

```
apimcli remove-env -n <environment-name>
apimcli remove-env --name <environment-name>
```

1. Make sure that WSO2 API Manager is started and that the CLI import/export tool is running.



- Run the following CLI command to list the environments.

There are no flags for the following CLI command.

```
apimcli list envs
```

Migrating APIs to different environments

Exporting an API

- Make sure that WSO2 API Manager is started and that the CLI import/export tool is running.
- Run the following CLI command to export an existing API as a .zip archive.

```
apimcli export-api -n <API-name> -v <version> -r <provider> -e  
<environment> -u <username> -p <password> -k  
apimcli export-api --name <API-name> --version <version> --provider  
<provider> --environment <environment> --username <username> --password  
<password> --insecure
```

The username and password are optional flags. You will be prompted to enter your credentials if you do not provide these flags.

**Flags:**

Required flags:

```
--name, -n  
--version, -v  
--provider, -r  
--environment, -e  
--insecure, -k : This allows connections to SSL sites without  
certificates
```

Optional flags:

```
--username, -u  
--password, -p
```

Importing an API

When you import an API, regardless the status of the imported API it will be added with the created state and you need to sign in to the Publisher and publish the API.

You can use the archive created in the previous section to import APIs to an API Manager instance.



1. Make sure that WSO2 API Manager is started and that the CLI import/export tool is running.

#### For Secure Endpoint Enabled APIs:

If you have enabled secure endpoints when creating the API and your username or/and password differs in the two environments, please follow the steps below before importing the API.

1. Unzip the .zip archive created in the previous section.
  2. Go to the <API-name-version>/Meta-information directory and open the api.json file.  
For example, PhoneVerification\_1.0.0/Meta-information directory and open the api.json file.
  3. Modify the endpointUTPassword with your endpoint password and save the api.json file.
  4. Compress the PhoneVerification\_1.0.0 folder to a folder named myExportedAPI.
- Run the following CLI command to import an API.

```
apimcli import-api -f <environment>/<file> -e <environment> -u <username> -p <password> -k  
apimcli import-api --file <environment>/<file> --environment <environment> --username <username> --password <password> --insecure
```

The username and password are optional flags. You will be prompted to enter your credentials if you do not provide these flags.

#### Flags:

Required flags:

--file, -f : The file path of the exported API. For example, if your file path is /Users/kim/.wso2apimcli/exported/apis/dev/PhoneVerification\_1.0.0.zip., then you need to enter dev/PhoneVerification\_1.0.0.zip as the value for this flag.

--environment, -e : The environment to which you want to import the API to.

--insecure, -k : This allows connections to SSL sites without certificates

Optional flags:



```
--username, -u
```

```
--password, -p
```

You must add a parameter named `--preserve-provider` to the CLI command and set its value to false if the API is imported to a different domain than its exported one. This parameter sets the provider of the imported API to the user who is issuing the CLI command. Here's an example:

```
apimcli import-api -k -f <environment>/<file> -e <environment> -u  
<username> -p <password> --preserve-provider <preserve_provider>  
apimcli import-api --insecure --file <environment>/<file> --environment  
<environment> -u <username> -p <password> --preserve-  
provider=<preserve_provider>
```

The username and password are optional flags. You will be prompted to enter your credentials if you do not provide these flags.

#### Flags:

Required flags:

```
--insecure, -k : This allows connections to SSL sites without  
certificates
```

```
--file, -f
```

```
--environment, -e : The environment to which you want to import  
the API to.
```

```
--preserve-provider (This does not have a short flag)
```

Optional flags:

```
--username, -u
```

```
--password, -p
```

The `--preserve-provider` flag is used to decide whether to keep the actual Provider as the provider of the API or change the provider to the user who is importing the API to a different environment.

As an example, If `--preserve-provider` is set to true, when importing an API created by user-1 in environment-1 will be preserved as the provider when and after importing that API to environment-2 by user-2. If `--preserve-provider` is set to false, when importing that API created by user-1 to the environment-2, the provider will be changed (not preserved) to user-2 who is importing the API.

## Troubleshooting

After importing, if the APIs are not visible in the API Publisher UI, do the following to re-index the artifacts in the registry.

Rename the `<lastAccessTimeLocation>` element in the `<API-M_2.6.0_HOME>/repository/conf/registry.xml` file. If you use a **clustered/distributed API Manager setup**, change the file in the API Publisher node. For example, change the `/_system/local/repository/components/org.wso2.carbon.registry/indexing/lastaccesstime` registry path to `/_system/local/repository/components/org.wso2.carbon.registry/indexing/lastaccesstime_1`.

1. Shut down the API Manager 2.6.0, backup and delete the `<API-M_2.6.0_HOME>/solr` directory.

For more information, see [Upgrading the API Manager to 2.6.0](#).



# Demo 9 – Microgateway Quick Start

Before you begin,

1. Download the [Microgateway](#) distribution and extract it.
2. Append the full path of the /bin folder of the extracted Microgateway distribution to the PATH environment variable.

If you are unable to append the path, you can alternatively run the Microgateway commands by navigating to the <MICROGW\_HOME>/bin folder and running the commands as ./micro-gw instead.

Let's go through the main use case of the Microgateway:

Deploy and subscribe to the sample API

1. Open the API Publisher (<https://<hostname>:9443/publisher>) and sign in with admin/admin credentials.
2. Close the interactive tutorial that starts automatically if you are a first-time user.
3. Click the **Deploy Sample API** button. It deploys a sample API called PizzaShackAPI into the API Manager.  
The **Deploy Sample API** option is available only when there are no APIs in API Publisher. If you have already created a API, this option is not available.
4. Click PizzaShackAPI to open it.
5. Go to the **Lifecycle** tab and note that the **State** is PUBLISHED. The API is already published to the API Store.
6. Sign in to the API Store (<https://<hostname>:9443/store>) with the admin/admin credentials and click the PizzaShackAPI API.
7. Select the default application and an available tier, and click **Subscribe**.
8. When the subscription is successful, click **View Subscriptions** on the information message that appears.

You have now successfully subscribed to an API. Let's generate a distribution for this API.

Generate a distribution for a single API

It is possible to create a Microgateway distribution for a single API that is in the **Published** state.

For details on how to create a Microgateway distribution for a group of APIs, see [Grouping APIs with Labels](#).



1. Navigate to a preferred workspace folder using the command line.  
This location is used to run the Microgateway commands and to generate Microgateway artifacts.

2. Set up a project using the command given below,

```
micro-gw setup <project_name> -a <API_name> -v <version>
```

3. When the tool requests for the username and password, use admin in both instances.  
Use the default values for the other parameters by pressing **Enter**.

4. Here is an example:

```
$ micro-gw setup pizzashack-project -a PizzaShackAPI -v 1.0.0
Enter Username:
admin
Enter Password for admin:

Enter APIM base URL [https://localhost:9443/]:

Enter Trust store location:
[lib/platform/bre/security/ballerinaTruststore.p12]

Enter Trust store password: [ use default? ]

Setting up project pizzashack-project is successful.
```

5. When the above command is issued, the tool connects with API Manager REST APIs and retrieves the API specified above. The source artifacts are generated in the current folder location.

6. The folder structure looks similar to the following,

```
.
└── pizzashack-project
    ├── conf
    │   └── deployment-config.toml
    └── src
        ├── extension_filter.bal
        ├── listeners.bal
        ├── PizzaShackAPI_1_0_0.bal
        └── policies
            ├── application_10PerMin.bal
            ├── application_20PerMin.bal
            ├── ...
            ├── subscription_Bronze.bal
            ├── subscription_Gold.bal
            ├── ...
            └── subscription_Unauthenticated.bal
```

```
|   └── throttle_policy_initializer.bal  
└── target
```

7. If you re-run the setup command while you already have a project named `pizzashack-project` in the current working directory, you receive an error similar to the following:

```
$ micro-gw setup pizzashack-project -a PizzaShackAPI -v 1.0.0  
  
[micro-gw: Project name `pizzashack-project` already exist. use -f or --force to forcefully update the project directory., Run 'micro-gw help' for usage.]
```

8. As suggested in the above output, if you need to override the current project, run the setup command with the `-f` (`--force`) option.

```
$ micro-gw setup pizzashack-project -a PizzaShackAPI -v 1.0.0 -f  
  
Enter Password for admin:  
  
Setting up project pizzashack-project is successful.
```

9. Build the microgateway distribution for the project using the following command:

```
micro-gw build <project_name>
```

10. Here is an example:

```
$ micro-gw build pizzashack-project  
  
Build successful for the project - pizzashack-project
```

11. Once the above command is executed, the generated source files are built and a Microgateway distribution is created in the target folder.

12. The name of the distribution will have the format `micro-gw-<project_name>.zip`. In the above example, the name will be `micro-gw-pizzashack-project.zip`.
13. Next, unzip the `micro-gw-pizzashack-project.zip` and run the gateway shell script inside the `bin` folder of the extracted zip using the following command:

```
bash gateway
```

14. The Microgateway will now start for the Pizzashack API. For example,

```
micro-gw-pizzashack-project/bin$ bash gateway

ballerina: HTTP access log enabled
ballerina: initiating service(s)
in '/home/user/workspace/pizzashack-project/target/micro-gw-
pizzashack-project/exec/pizzashack-project.balx'
ballerina: started HTTPS/WSS endpoint 0.0.0.0:9096
ballerina: started HTTPS/WSS endpoint 0.0.0.0:9095
ballerina: started HTTP/WS endpoint 0.0.0.0:9090
```

You have successfully generated and started a Microgateway distribution for the Pizzashack API.

Generate a JWT token and invoke the API

Once you start the Microgateway, you can use a JWT or an OAuth2 token to invoke the API. For the invocation URL, you can use

either `https://localhost:9095/` (HTTPS)  
or `http://localhost:9090/` (HTTP).

A signed JWT token is a self-contained access token that can be validated by the Microgateway itself without connecting to any external party. In contrast, a typical OAuth2 token needs to be validated by connecting to the Key Manager. This is one of the key advantages of the Microgateway, if you need to run it in offline mode.

1. Log in to the API Store, click **Applications** and choose to edit the Default Application. Set the token type to JWT.

## Edit Application

Name\* DefaultApplication  
Characters left: 52

Per Token Quota Unlimited Allows unlimited requests  
This feature allows you to assign an API request quota per access token. The allocated quota will be shared among all the subscribed APIs of the application.

Description

Token Type JWT

**Info!**  
JWT token type can be used with micro gateways only

**Update** Cancel

2. Click the **Production Keys** tab and click **Generate Keys** to generate an access token to invoke the API.

If you have already generated keys before, click **Regenerate**.

3. Invoke the API using the JWT token using the following command:

### Format

```
curl -k -i -H "Authorization: Bearer <JWT_token>" <API_url>
```

For example,

### Example

```
curl -k -i -H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6Ii1VCX0JReTJIR1YzRU1UZ3E2NFEtMVZpdFliRSJ9.eyJhdWQiOiJodHRwOlwvXC9vcmcud3NvMi5hcGltZ3RcL2dhGv3YXkiLCJzdWIiOiJhZG1pbisImFwcGxpY2F0aW9uIjp7ImlkIjoxLCJuYW1lIjoiRGVmYXVsdfEFwcGxpY2F0aW9uIiwidGllciI6IlVubGltaXR1ZCJ9LCJzY29wZSI6ImFtX2FwcGxpY2F0aW9uX3Njb3BlIGRlZmF1bHQiLCJpc3MiOiJodHRwczpcL1wvbG9jYWxob3N
```

```
00jgyNDNcL3Rva2VuIiwia2V5dHlwZSI6I1BST0RVQ1RJT04iLCJzdWJzY3JpYmVkQVB
JcyI6W3sibmFtZSI6I1BpenphU2hhY2tBUEkiLCJjb250ZXh0IjoiXC9waXp6YXNoYWN
rXC8xLjAuMCIsInZlcnNpb24iOiIxLjAuMCIsInB1Ymxpc2hlciI6ImFkbWluIiwic3V
ic2NyaXB0aW9uVGllciI6I1VubGltaXR1ZCIsInN1YnNjcmliZXJUZW5hbnREb21haW4
i0iJjYXJib24uc3VwZXIfV0sImV4cCI6MTUyODgwMjg3ODExMCwiaWF0IjoxNTI4ODA
yODgwMTEwLCJqdGki0iJlZmQxZGVhZC1iNWI3LTQ0M2MtOTIyMC1iYzJjZTJjY2MwYjE
ifQ==.b8P2uPRkVai007PcbvbjANLuH1JQzX1eHplweDpE6ItbEHRTkN2U_h6b39tz14
dKUmigzASinj5LheUWGB7gEDRqlc39ckhRX2qpolQpITZvpzYo8ky9Acx1JXLxrfPwgd
ht36zfIQwlPN_s2A5nY7c9pDBMu00001YpmK81SrtipFSTAyPiRg5VyY3n-4POnjkEF-
LQKCTq7ef0uLOFTcoCT-gqNsXeKqt15suCYj5QMHJ8VP5bKsKZy9-
1o9oFNlwc1QE0qE01fPuGuz-
4J220vkHyrasbjhhGaaDgdpdER19ElUDuL0C9AdX6Fb1sz54gnAiU3RUBK3RQUDK7Q==
" https://localhost:9095/pizzashack/1.0.0/menu
```

You receive a response similar to the following:

```
HTTP/1.1 200 OK
Date: Thu, 26 Jul 2018 16:52:38 GMT
Content-Type: application/json
Transfer-Encoding: chunked
server: WS02 Carbon Server

[{"name": "BBQ Chicken Bacon", "description": "Grilled white chicken, hickory-smoked bacon and fresh sliced onions in barbecue sauce", "price": "13.99", "icon": "/images/6.png"}, {"name": "Chicken Parmesan", "description": "Grilled chicken, fresh tomatoes, feta and mozzarella cheese", "price": "9.99", "icon": "/images/1.png"}, {"name": "Chilly Chicken Cordon Bleu", "description": "Spinash Alfredo sauce topped with grilled chicken, ham, onions and mozzarella", "price": "17.99", "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": "16.99", "icon": "/images/9.png"}, {"name": "Garden Fresh", "description": "Slices onions and green peppers, gourmet mushrooms, black olives and ripe Roma tomatoes", "price": "9.99", "icon": "/images/3.png"}, {"name": "Grilled Chicken Club", "description": "Grilled white chicken, hickory-smoked bacon and fresh sliced onions topped with Roma tomatoes", "price": "13.99", "icon": "/images/8.png"}, {"name": "Hawaiian BBQ Chicken", "description": "Grilled white chicken, hickory-smoked bacon, barbeque sauce topped with sweet pine-apple", "price": "27.99", "icon": "/images/7.png"}, {"name": "Spicy Italian", "description": "Pepperoni and a double portion of spicy Italian"}]
```

```
sausage", "price": "13.99", "icon": "/images/2.png"}, {"name": "Spinach Alfredo", "description": "Rich and creamy blend of spinach and garlic Parmesan with Alfredo sauce", "price": "27.99", "icon": "/images/5.png"}, {"name": "Tuscan Six Cheese", "description": "Six cheese blend of mozzarella, Parmesan, Romano, Asiago and Fontina", "price": "11.99", "icon": "/images/4.png"}]]
```

# Demo 10 – Deploying the API Microgateway in Kubernetes

1. Start the API Manager and log in to the API Publisher (<https://<hostname>:9443/publisher>) using `admin` as the username and password.
2. Create an API with the following details:

Field	Sample value
Name	hello_world - v1
Context	/hello/v1
Version	1.0.0
Access Control	All
Visibility on Store	Public
Production URL	<a href="http://bk.test.com">http://bk.test.com</a>
Tier Availability	Gold, Unlimited

3. You can create an application that supports JWT tokens, subscribe to the API and get a JWT token to invoke the API.
4. Create a `deployment.toml` file enabling Kubernetes deployment, service and config map resources.

The config map is used to copy the `micro-gw.conf` file.

```
[kubernetes]
[kubernetes.kubernetesDeployment]
enable = true
#name = ''
#labels = ''
#replicas = ''
#enableLiveness = ''
#initialDelaySeconds = ''
#periodSeconds = ''
#livenessPort = ''
#imagePullPolicy = ''
#image = ''
```

```

#env = ''
#buildImage = ''
#copyFiles = ''
#dockerHost = ''
#dockerCertPath = ''
#push = ''
#username = ''
#password = ''
#baseImage = ''
#singleYAML = ''
[kubernetes.kubernetesService]
enable = true
#name = ''
#labels = ''
serviceType = 'NodePort'
#port = ''
[kubernetes.kubernetesConfigMap]
enable = true
ballerinaConf = '/home/user/wso2am-micro-gw-toolkit-
2.5.0/resources/conf/micro-gw.conf'
#[[kubernetes.kubernetesConfigMap.configMaps]]
#    name = ''
#    mountPath = ''
#    readOnly = false
#    data = ['']

```

Let's create a project called `k8s_project` and provide the `deployment.toml` file as an input.

5. Navigate to the `wso2am-micro-gw-toolkit-2.5.0/bin` directory and run the following command,

```
./micro-gw setup k8s_project -a hello_world -v v1 --deployment-config deployment.toml
```

6. This command creates the following folders under the `k8s_project` folder.

```

├── k8s_project
|   ├── conf
|   |   └── deployment-config.toml
|   ├── src
|   |   ├── extension_filter.bal
|   |   ├── hello_world_v1.bal
|   |   ├── listeners.bal
|   |   └── policies
|   |       └── application_10PerMin.bal

```

```
|   |   └── application_20PerMin.bal  
|   |   └── application_50PerMin.bal  
|   |   └── subscription_Bronze.bal  
|   |   └── subscription_Gold.bal  
|   |   └── subscription_Silver.bal  
|   |   └── subscription_Unauthenticated.bal  
|   |   └── throttle_policy_initializer.bal  
|   └── target  
└── temp  
    └── hashes.json
```

7. Build the project using the following command,

```
./micro-gw build k8s_project
```

8. This generates the following Kubernetes resources.

```
└── k8s_project  
    ├── docker  
    │   └── Dockerfile  
    ├── k8s_project_config_map.yaml  
    ├── k8s_project_deployment.yaml  
    └── k8s_project_svc.yaml
```

9. The docker image to be deployed in Kubernetes is created in your local registry. You can find the image `k8s_project:latest` when you execute the `docker images` command.

10. Deploy the docker image in a Kubernetes environment.

You can push the docker image to the docker hub or to a private docker registry. If you change the docker image name, you need to change the image name in the `k8s_project_deployment.yaml` file.

11. You can also SCP the image to the Kubernetes nodes as follows:

- a. Save the docker image to a `.tar` file. For example,

```
docker save k8s_project:latest > image.tar
```

b. SCP the image to the Kubernetes nodes. For example,

```
scp -I <identity file> image.tar username@K8s_NODE_IP:
```

c. Load the docker image in the Kubernetes nodes. This needs to be executed in the Kubernetes nodes. For example,

```
docker load < image.tar
```

12. Deploy the API Microgateway in Kubernetes by deploying the Kubernetes resources using the following command,

```
kubectl create -f k8s_project/target/kubernetes/
```

13. Access the API in HTTPS using the following details:

The `NodePort` service type has been used in Kubernetes. For that service type, you can access the API using any of the Kubernetes node IP addresses and node ports.

`https://<Any_Kubernetes_Node_IP>:<NodePort>/hello/v1/check`

You can use the `kubectl get services` command to list down the services that run on Kubernetes.

Node port - 32616

URL - `https://<Any_Kubernetes_Node_IP>:32616/hello/v1/check`

Headers - `Authorization Bearer <JWT_TOKEN>`

Method - GET

As JWT is a self-contained access token, the Microgateway does not need to connect to the Key Manager. However, if you are using an OAuth2 access token, point the Microgateway to the Key Manager using the Key Manager details in the `micro-gw.conf` configuration file of the Microgateway. If you are running Key Manager in Kubernetes, you can provide the Key Manager `serverUrl` as shown below. The `serverUrl` has to be accessible from the Microgateway pods.

```
[keyManager]
```



```
serverUrl="https://localhost:9443"  
username="admin"  
password="admin"  
tokenContext="oauth2"  
timestampSkew=5000
```

