

Practical Exercise: Create a WebSocket API

# Training Objective

Learn how to create and use WebSocket APIs

# Business Scenario

PizzaShack wants to implement a WebSocket API.

# High-Level Steps

* Create and Publish a WebSocket API
* Use the wscat WebSocket client to invoke the API

# Detailed Instructions

# **Create and Publish a WebSocket API**

This tutorial will guide you to implement a WebSocket based chat application using the WSO2 API Manager. Follow the instructions in this tutorial to design and publish API via a WebSocket backend, and thereafter invoke the WebSocket API using the **wscat** WebSocket client.

This will demonstrate a simple command line based chat room which has two channels: **rooms**, and **notifications**.

**Note**

When you create a WebSocket Streaming API it's exposed via both ws and wss protocols. By default, the ws transport uses port 9099, and the wss transport uses port 8099.

### **Step 1 - Design a WebSocket API**

1. Sign in to the Publisher.

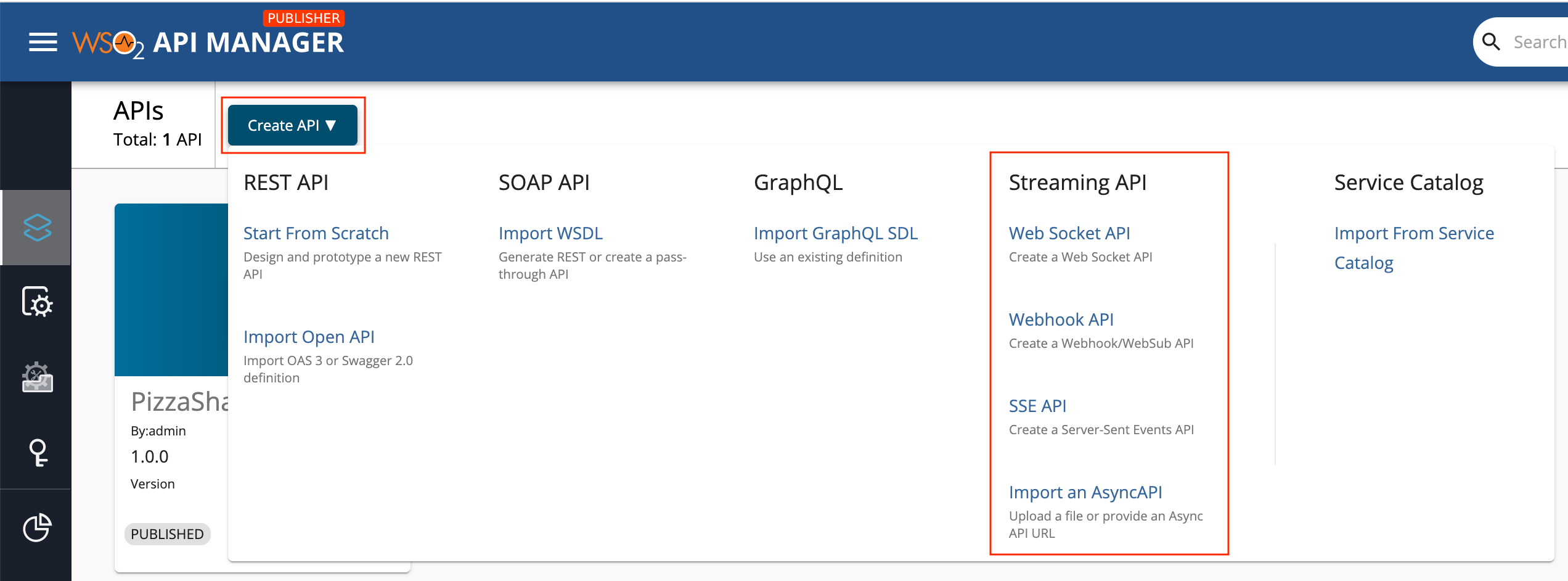
https://<hostname>:9443/publisher

For testing purposes, you can use https://localhost:9443/publisher and admin as the username and password.

1. Click **Create API**, go to **Streaming API**, and click **WebSocket API**.

**Note**

The **Create** button will only appear for a user who has the creator role permission.

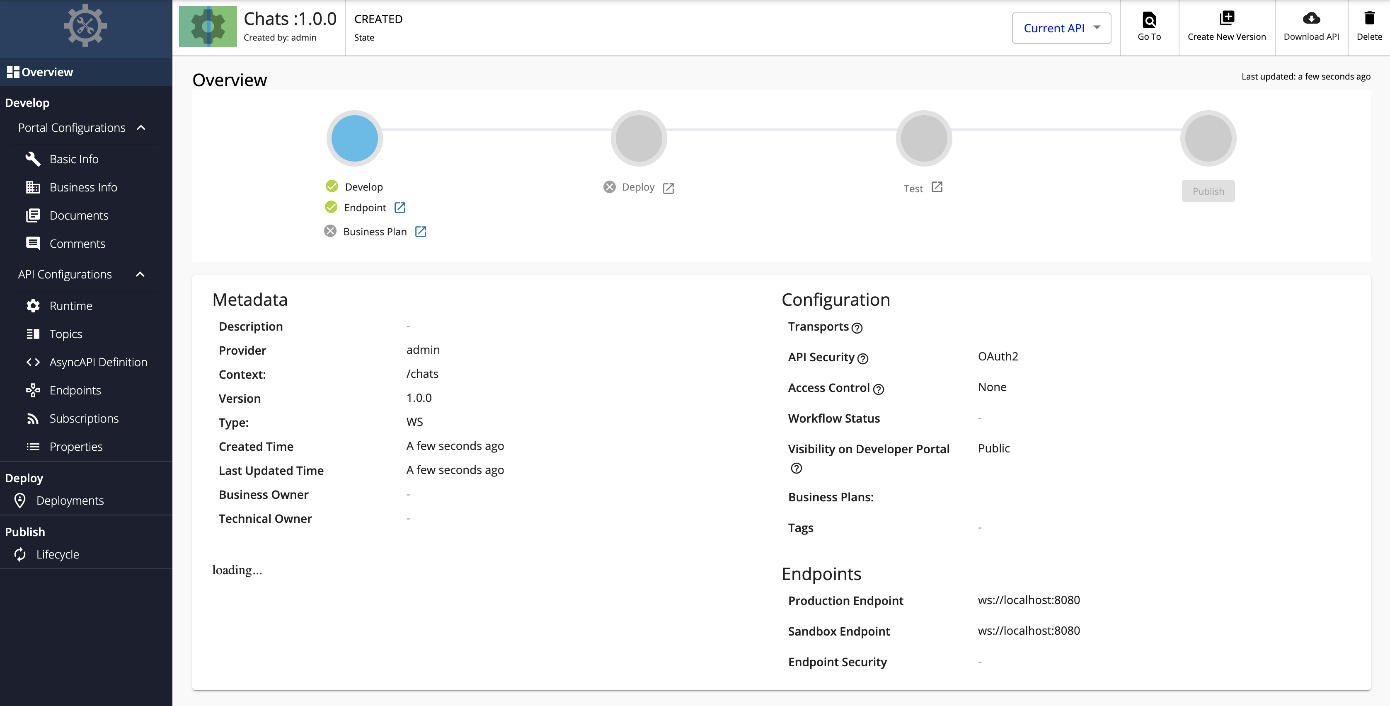
[](https://apim.docs.wso2.com/en/4.2.0/assets/img/design/create-api/streaming-api/design-new-streaming-api.png)

1. Enter the details of the new WebSocket API.

| **Field** | **Sample value** |
| --- | --- |
| Name | Chats |
| Context | /chats |
| Version | 1.0.0 |
| Protocol | WebSocket |
| Endpoint | ws://localhost:8080 |

1. **Note**
   * When you create a WebSocket Streaming API it's exposed via both ws and wss protocols. By default, ws transport uses port 9099, and wss transport uses port 8099.
   * For non-secured WebSockets enter the protocol as ws:// in the endpoint, or for secured WebSockets enter the protocol as wss://
2. Click **CREATE**.

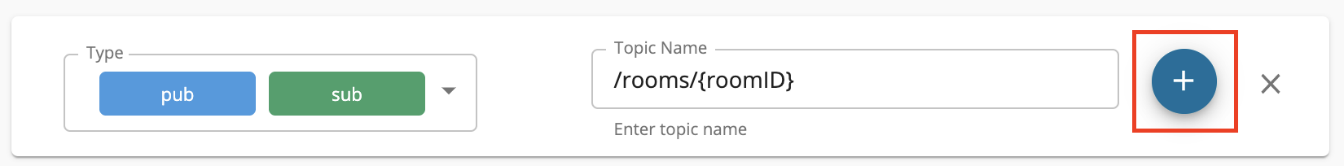
The overview page of the created WebSocket API appears.

[](https://apim.docs.wso2.com/en/4.2.0/assets/img/tutorials/streaming-api/websocket-api-overview.png)

1. Add topics to the WebSocket API.
   * Click **Topics** and navigate to the Topics page.
   * Delete the existing default topic, which has the name /\*.
   * Add the following topics one by one.

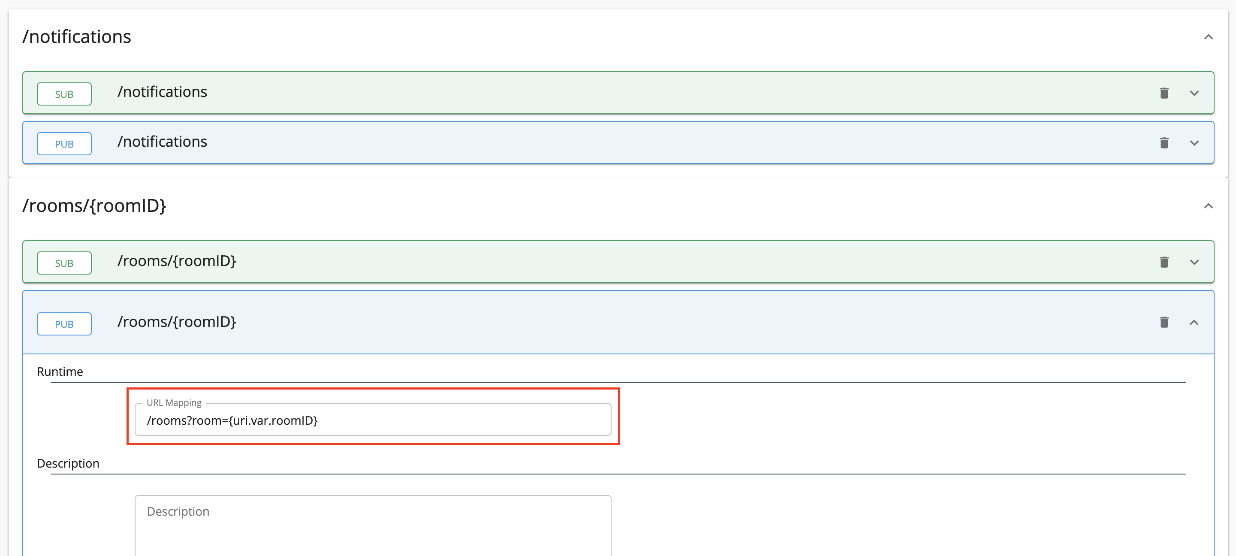
Select **pub** and **sub** as the **Types**, enter the **Topic Name**, and click **+** to add each topic.

* + - /notifications
    - /rooms/{roomID}

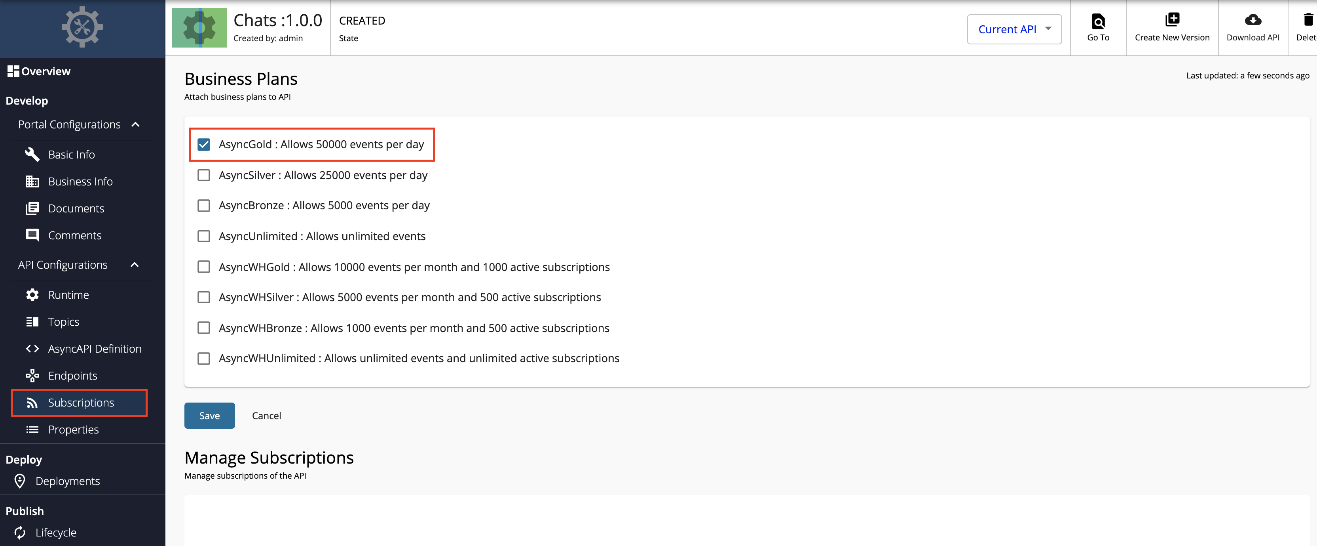


* + Expand each topic, provide URL Mappings as follows, and click **Save**.

|  |  |
| --- | --- |
| **Topic** | **URL Mapping** |
| /notifications | /notifications |
| /rooms/{roomID} | /rooms?room={uri.var.roomID} |

* + URL Mapping provided for a topic will be appended to the WebSocket endpoint URL, which was provided when creating the API, and the traffic via the topic will be sent to & received from the resulting URL.
  + [](https://apim.docs.wso2.com/en/4.2.0/assets/img/tutorials/streaming-api/websocket-api-topic-url-mapping.png)

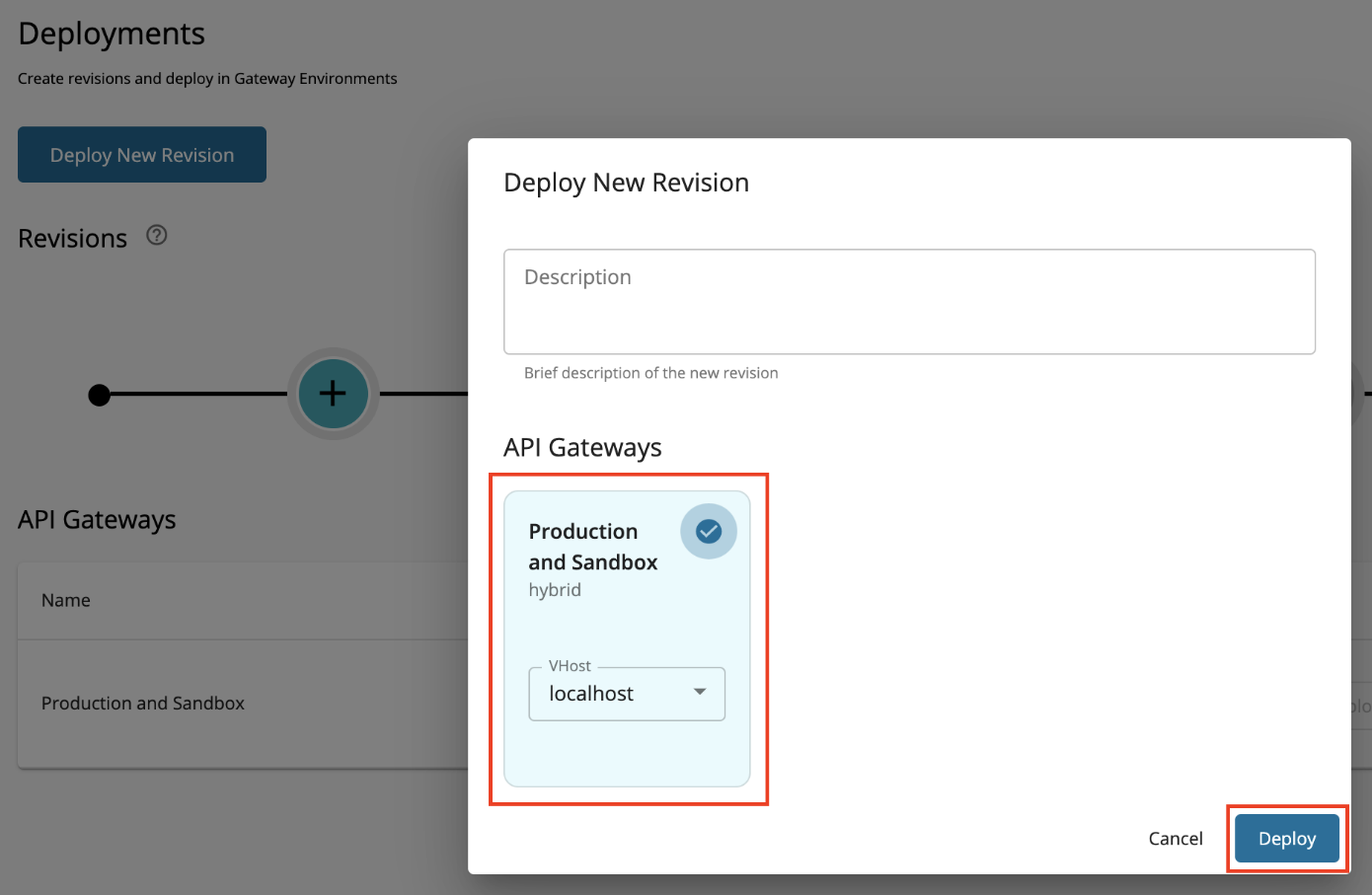
1. Attach business plans to your WebSocket API.
   * Click **Subscriptions** and navigate to the Business Plans page.
   * Select **AsyncGold** and click on **Save**.

[](https://apim.docs.wso2.com/en/4.2.0/assets/img/tutorials/streaming-api/websocket-api-subscriptions.png)

Now, you have created and configured the WebSocket API successfully.

### **Step 2 - Publish the WebSocket API**

1. Click **Lifecycle** to navigate to the API lifecycle, and click **Publish** to publish the API to the API Developer Portal.
2. Click **Deployments** to navigate to the Deployments page and click **Deploy New Revision**.
3. Select **Production and Sandbox**, choose **localhost** as the VHost, and click on **Deploy**.

[](https://apim.docs.wso2.com/en/4.2.0/assets/img/tutorials/streaming-api/streaming-api-deploy-new-revision.png)

### **Step 3 - Start the WebSocket Server**

1. Download the sample WebSocket server from [WSO2 APIM Samples - GitHub repository](https://github.com/wso2/samples-apim/tree/master/streaming-api-backends/websocket-backend).
2. Go to the streaming-api-backends/websocket-backend directory, and install the required dependencies.

npm install

1. Start the server.

npm start

### **Step 4 - Invoke the WebSocket API**

1. Sign in to the Developer Portal.

https://<hostname>:9443/devportal

For testing purposes, you can use https://localhost:9443/devportal and admin as the username and password.

1. Click on the WebSocket API.

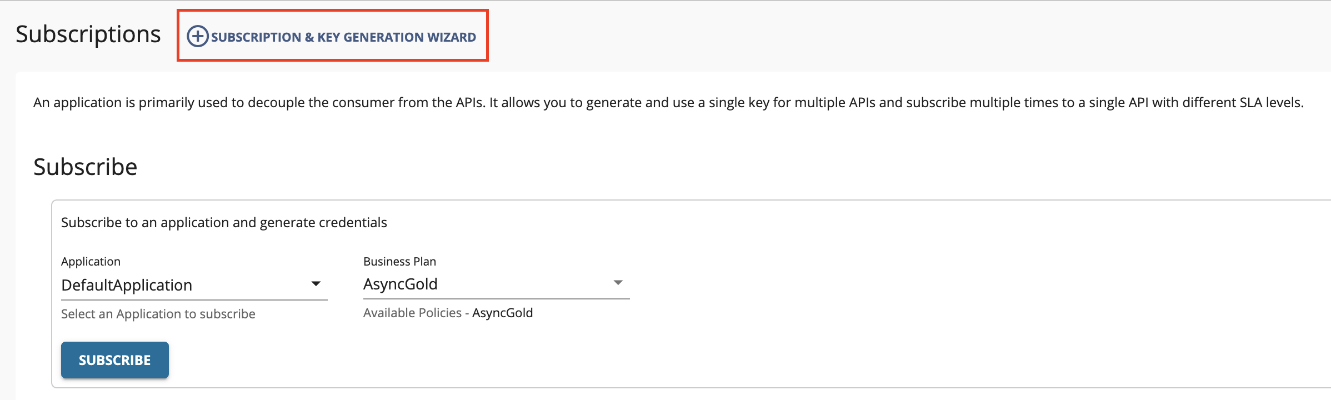
The API overview appears.

1. Subscribe to the API.
   1. Click **Subscriptions** to go to the Subscriptions page and click **Subscription & Key Generation Wizard**.

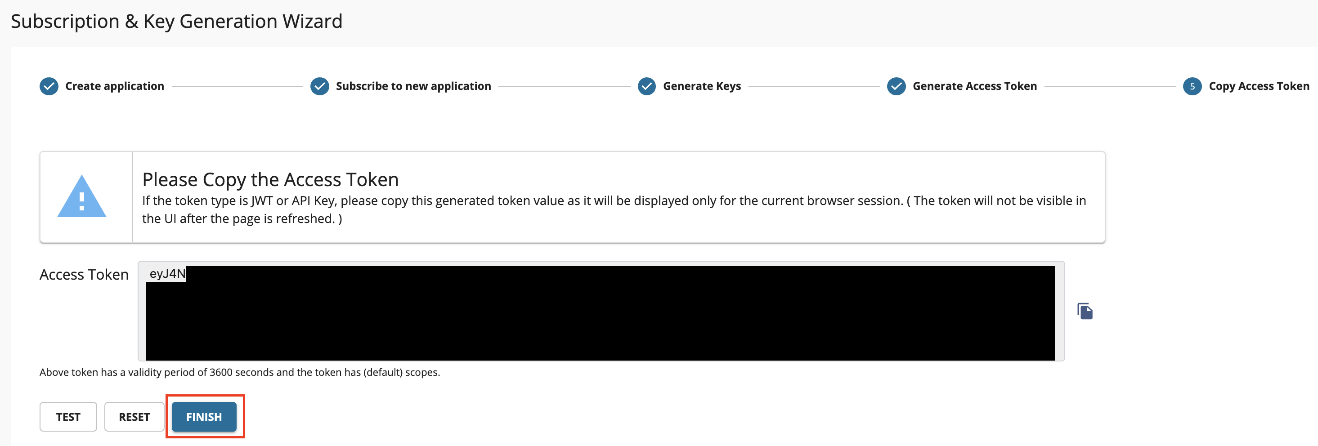
This wizard takes you through the steps of creating a new application, subscribing, generating keys, and generating an access token to invoke the API.

**Note**

You can use any application (e.g., JWT or OAuth) to subscribe to the API.

[](https://apim.docs.wso2.com/en/4.2.0/assets/img/tutorials/streaming-api/streaming-api-key-generation-wizard.png)

* 1. Copy the authorization token that appears, and click **Finish**.

[](https://apim.docs.wso2.com/en/4.2.0/assets/img/tutorials/streaming-api/streaming-api-subscription-token.png)

1. Try out the operations.
   1. Install **wscat** client.

npm install -g wscat

* 1. Invoke the API's /notifications topic with an authorization header by executing the following command.

WS

wscat -c ws://localhost:9099/chats/1.0.0/notifications -H "Authorization: Bearer [accesstoken]"

WSS

When the connection is successful, the WebSocket server will send:

Subscribed to notifications!

* 1. In a separate terminal, invoke the API's /rooms/{roomID} topic with an authorization header by executing the following command.

WS

wscat -c ws://localhost:9099/chats/1.0.0/rooms/room1 -H "Authorization: Bearer [accesstoken]"

WSS

When the connection is successful, the WebSocket server will send:

You joined room1!

This denotes that the first user has connected to room1.

Additionally, the following message will be shown in the terminal where you invoked the /notifications topic. This denotes the notification for the above event.

Someone joined room1!

* 1. In another terminal, invoke the API's /rooms/{roomID} topic again. This denotes the second user, who will be connecting to room1.

WS

wscat -c ws://localhost:9099/chats/1.0.0/rooms/room1 -H "Authorization: Bearer [accesstoken]"

WSS

You will receive the message: You joined room1! in this terminal, along with the corresponding notification in the notifications terminal.

As there are two users connected to room1, both of them can send and receive chats via room1. Try sending messages from both of these terminals back and forth.

**Note**

There are clients (especially browsers) that do not allow you to add headers to the WebSocket handshake. In such cases, you can send the access token for the WebSocket API invocation as a query parameter named access\_token by using the command below:

WS

wscat -c "ws://localhost:9099/chats/1.0.0/notifications?access\_token=[accesstoken]"

WSS

**Note**

BasicAuth and API Key do not work for the security of WebSocket APIs.

You have successfully created and published your first WebSocket API, subscribed to it, obtained an access token for testing, and tested your API with the access token.

 [CC BY 4.0](http://creativecommons.org/licenses/by/4.0/)