WSO2 API Manager 4.2.0 Fundamentals

Getting Started with the API Publisher

WSO2 Training

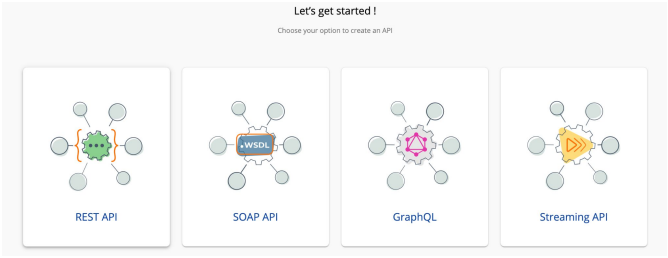
What is the API Publisher Portal?

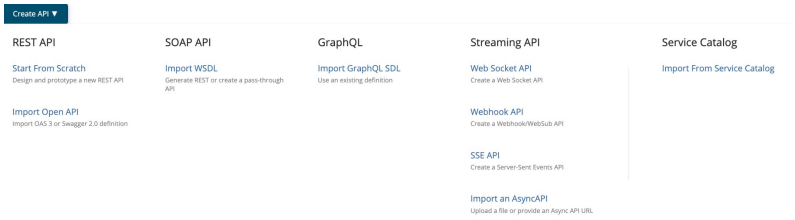
- React Web Application
- Story of APIs start here.
- Objectives
  - Design APIs
  - Deploy APIs
  - Publish APIs
  - Manage APIs

Creating an API

Ways of Creating APIs

- Design a new REST API
- REST API from OpenAPI or Swagger definition
- REST API from SOAP backend*
- SOAP API as a REST service*
- GraphQL API from SDL
- Design new WebSocket API
- Design Webhook/WebSub API
- Create SSE (Server-Sent Events) API
- Import an Async API
- Import from Service Catalog
- Add third party APIs





[Creating APIs]

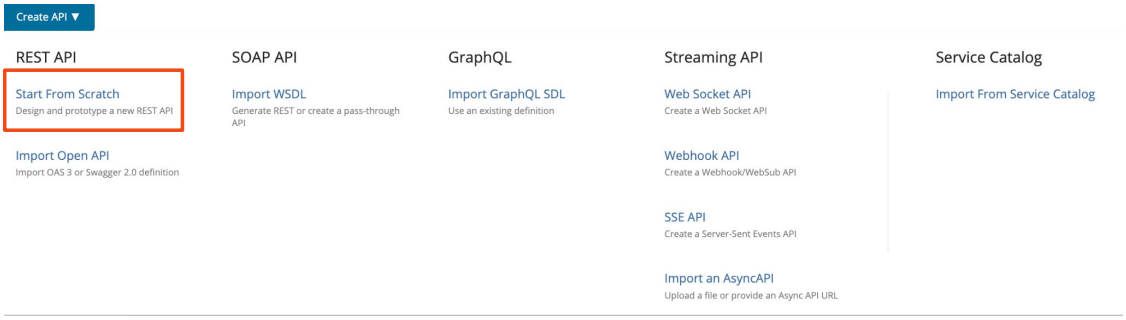
* SOAP APIs are deprecated in APIM 4.2.0. Therefore it is recommended to use WSO2 integration studio as an alternative. For more information please refer https://apim.docs.wso2.com/en/4.2.0/integrate/develop/creating-artifacts/creating-an-api/

Let’s try it out!

Creating and publishing an API through API Manager Publisher

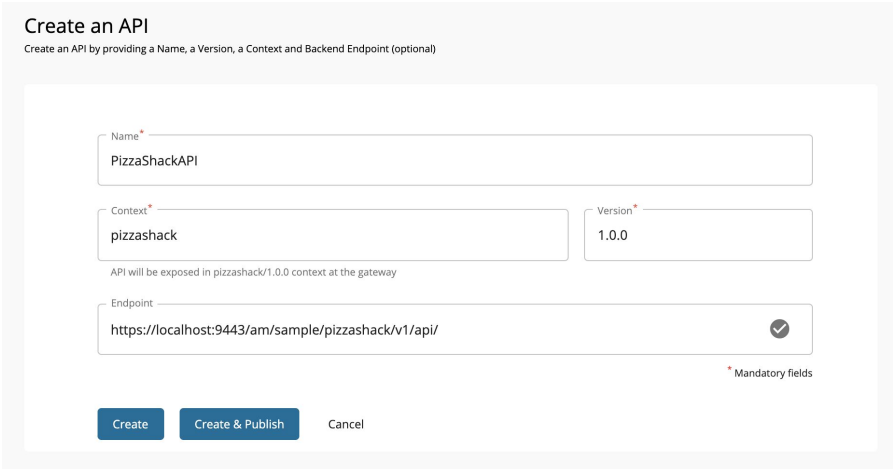
Design a New REST API

Select the **Start from Scratch** option



This section includes how to create a REST API from scratch and a walkthrough of the API Publisher UI to identify components and their functions.

Basic Details

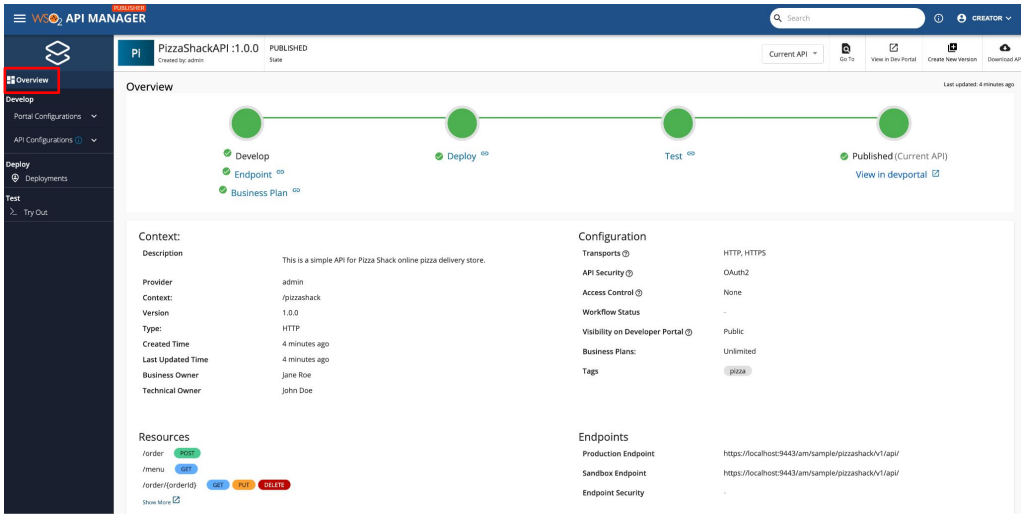


Provide the basic details for the API.

- **API Name** Name of the API is mandatory. Special characters and empty spaces are not allowed for API Name
- **Context** Root context of the API is mandatory.
- **Version** Version for the API is mandatory. For any API published via API Manager, API context + version must be unique.
- **Endpoint** Backend Endpoint of the API. This field is not mandatory.
- **Business Plan** Select the subscription policy/ policies the API should allow. This field is not mandatory

In this step, it is possible to provide all the information and publish the API in a single step, or create an API by only providing mandatory information.

API Overview



After creating the API, you will be redirected to **API Overview** page.

This is a summarized view of the details of the API.

API State Representation

- Represents the current state of the API
- Shows required actions to complete the current state.

Metadata

- Shows the basic information of the API.

Configuration

- Displays the basic configurations of the API. i.e., API security scheme, API Visibility information etc.

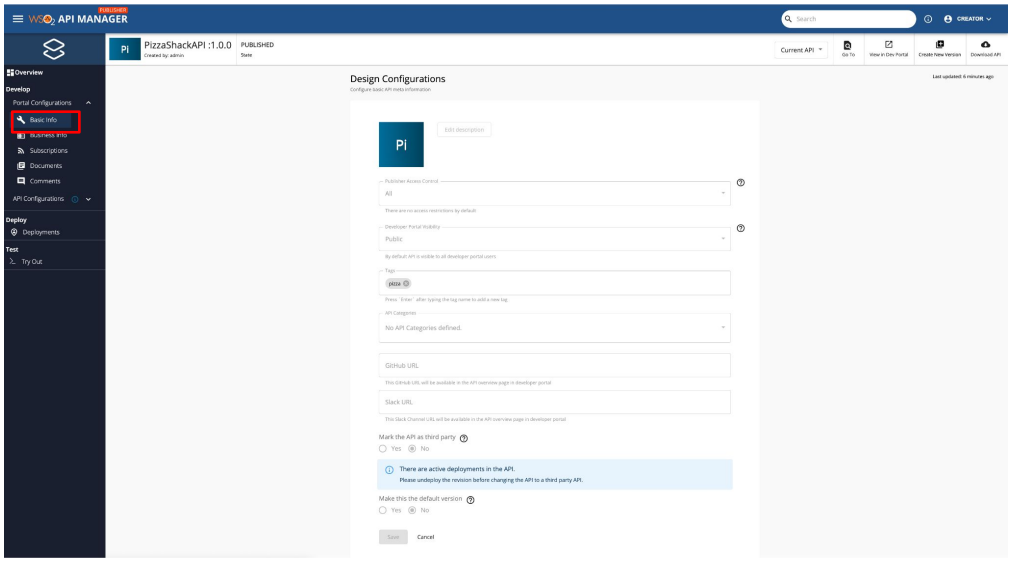
Resources

- List the Resources of the API

Endpoints

- Shows the production/ sandbox endpoints and whether they are secured.

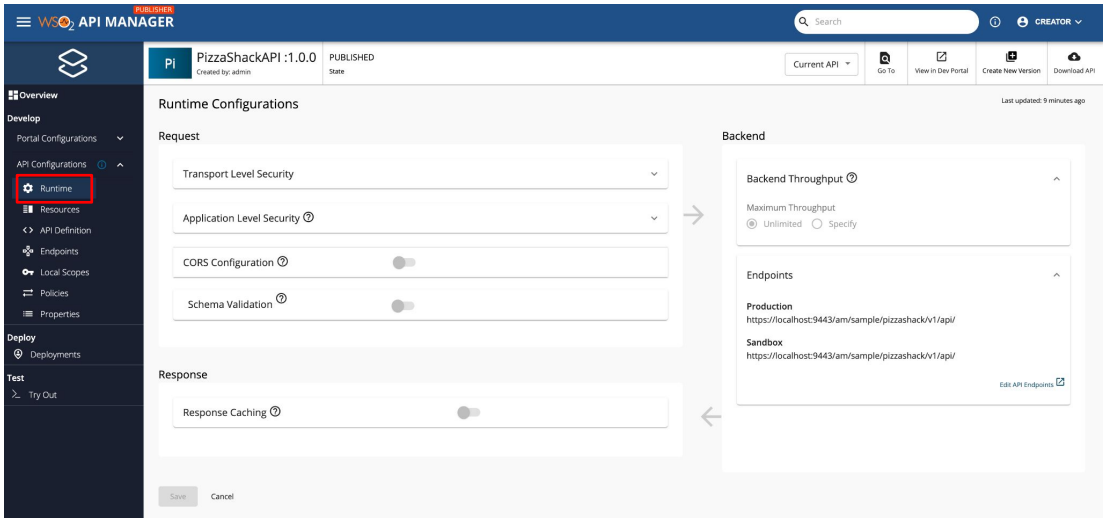
API Design Configurations



Design Configuration includes,

- Publisher Access control in the Publisher portal — To which roles, the API should be visible
- Developer Portal Visibility in the Developer Portal — To which roles the API should be visible
- Tags — Add tags to the API.
- API Categories — Set API categories
- GitHub URL
- Slack URL
- Mark this API as third party gateway — API will not be proxied through the
- Make the API the default version

API Runtime Configurations



API Runtime Configuration includes following sections

Request : How the request should be handled in the API.

- **Transport Level Security** Set the protocol type (HTTP/ HTTPS), enable Mutual SSL
- **Application Level Security** Set the application level security settings. (OAuth2, API Key)
- **CORS Configurations** Set Cross Origin Resource Sharing policies
- **Schema Validation** Set/ unset schema validation for requests

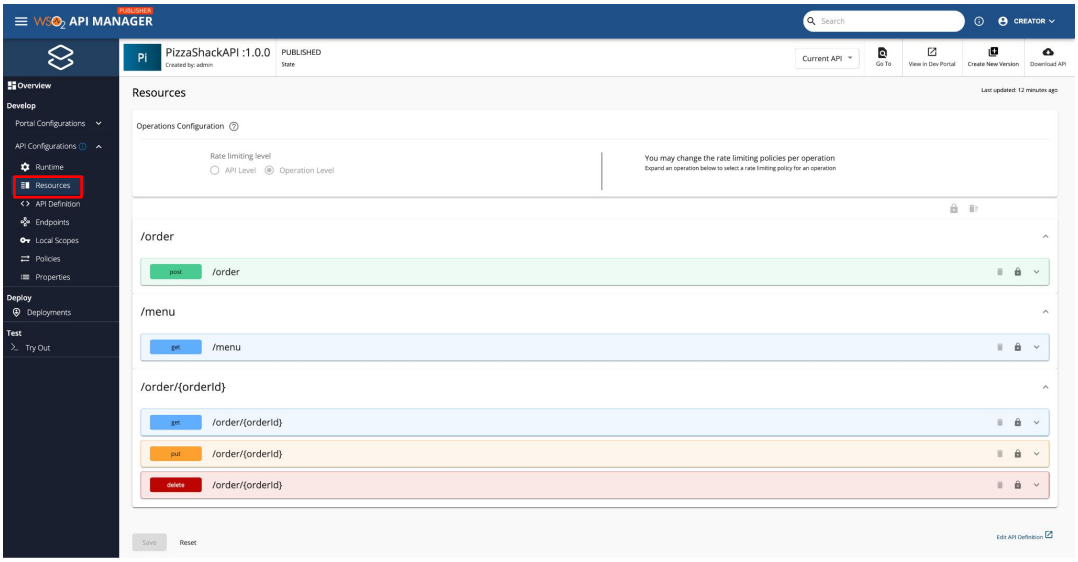
Response: How the Response flow should be handled.

- **Response Caching** Set Response caching configuration

Backend: The configuration related to the backend endpoint

- **Backend Throughput** Set the maximum TPS that the backend could handle
- **Endpoints** Displays the Production/ Sandbox backend endpoints.

API Resources



Add/ edit or remove resources from the API and set API level, resource level policies.

Supported HTTP Methods

- **GET**
- **POST**
- **PUT**
- **DELETE**
- **HEAD**
- **OPTION**

Delete a Resource Click on Delete in each resource. Once clicked, it will be marked as deleted.

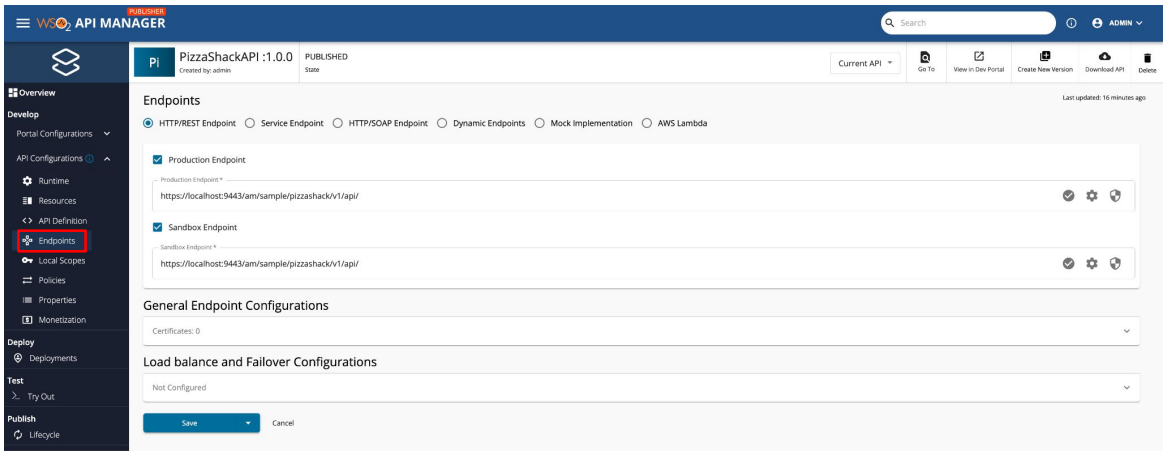
Add new resource

1. Select the HTTP method/ methods
2. Enter the resource path
3. Click “+”

Once added, it will be marked with a red dot.

To save the changes, (deletion/ addition) click Save.

Endpoints



Adding endpoints can be done via the Endpoints page.

If no endpoint is provided in the first step, this page will show an Endpoint Type selection.

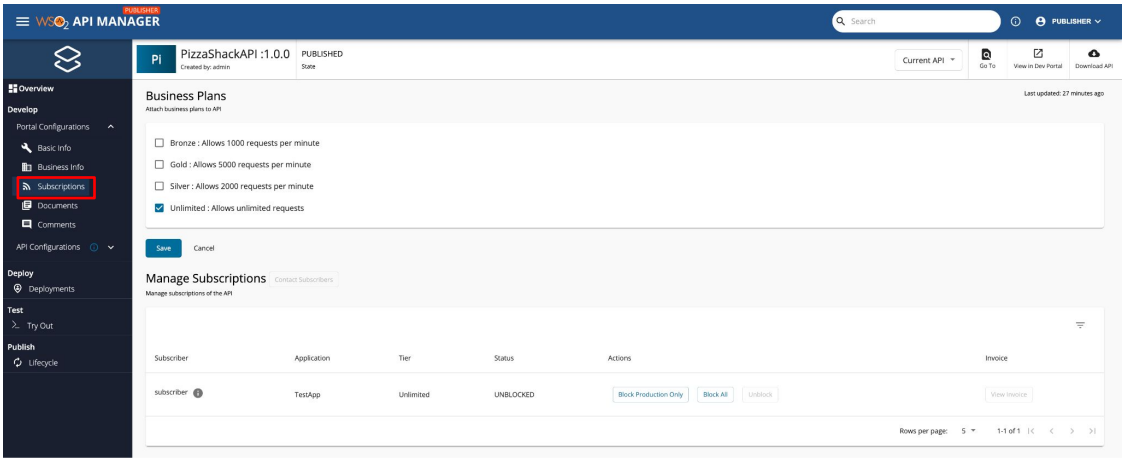
- **HTTP/ REST Endpoint** HTTP endpoint based on URI Template
- **HTTP/ SOAP Endpoint** Service URL of a SOAP web service. (for SOAP APIs)
- **Service Endpoint** Expose WSO2 Integration Studio integration as the backend for the API (https://apim.docs.wso2.com/en/4.2.0/tutorials/develop-an-integration-with-a-managed-api/#step-7-add-the-service-endpoint-to-the-api)
- **Mock Implementation** Use inbuilt javascript engine to mock the api response based on OpenAPI definition
- **Dynamic Endpoint** Route the request dynamically based on TO header.
- **AWS Lambda Endpoint** Invoke AWS Lambda functions directly through API

Once the endpoint type is selected, click ADD to add the endpoint to the API.

Endpoints Page

- **Endpoint Type** Change the endpoint type
- **Add Production/ Sandbox endpoints** Check the required endpoint type and enter the URL. Click **Check** in the input field to Test the health of the endpoint.
- **General Configuration** Configure Endpoint Security, and backend certificates
- **Load balance/ Failover Configuration** Add Load balance or Failover endpoints

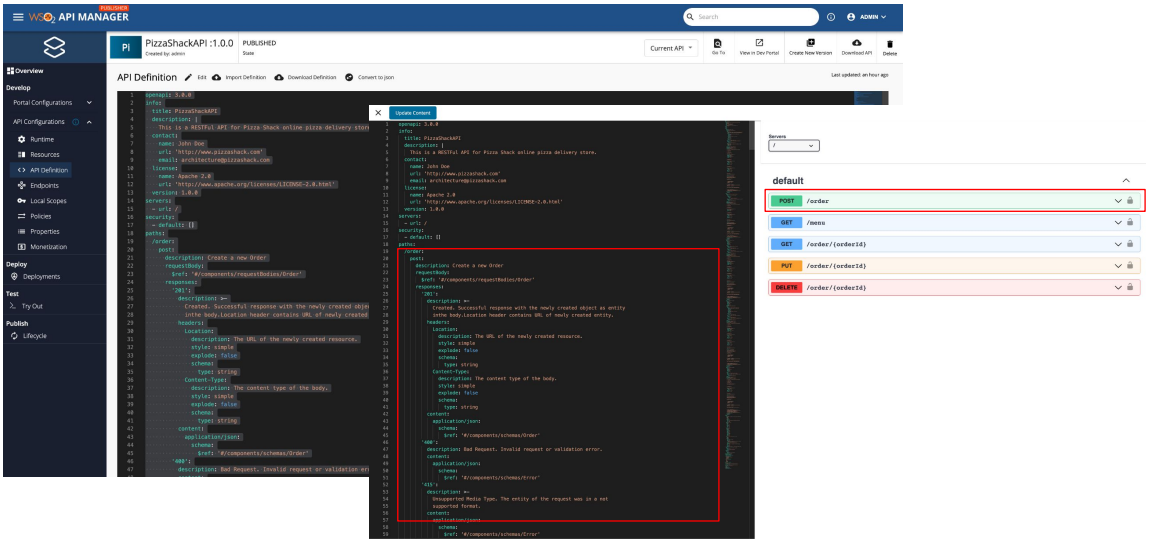
Business Plans and Subscriptions



Selecting business plan(s) for the API and view the consumer application subscriptions who has subscribed to this API

Also, this page also displays the existing API Subscriptions where we can manage them.

API Definition



View the auto-generated open API definition (Swagger) of the API.

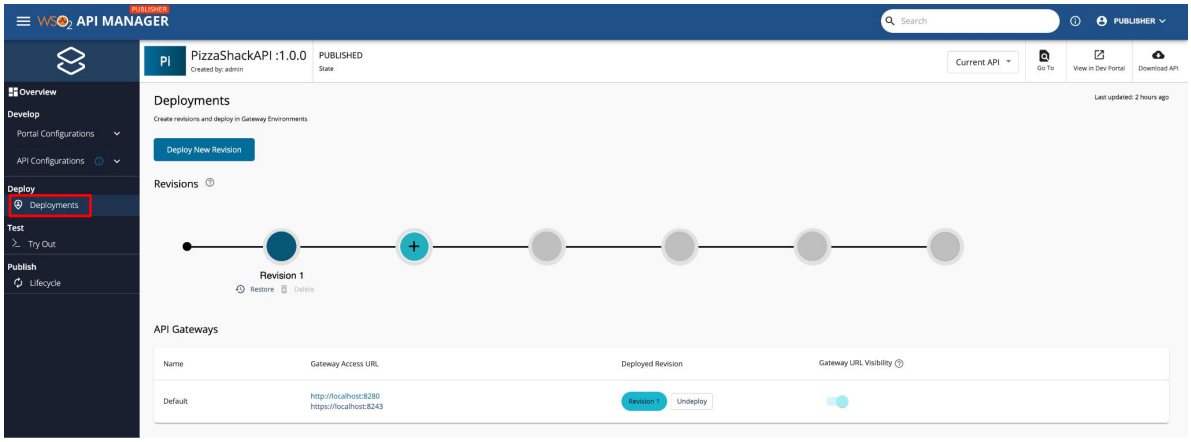
Edit the Definition

- To edit the definition in integrated Swagger Editor, click Edit.
- Do the necessary modifications. (add definition, rename existing paths, add schemas etc)
- Click Update Content to save the changes.

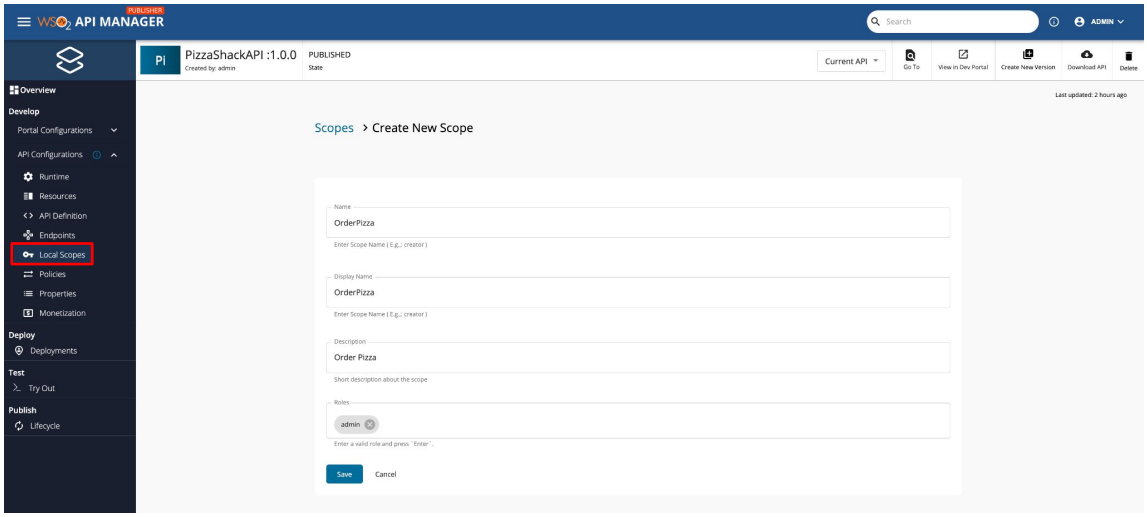
In this view you also can

- Download the definition in yaml or json format.
- Upload a new definition replacing the existing one
- Change the display format to json or yaml

Deployments and API Gateways



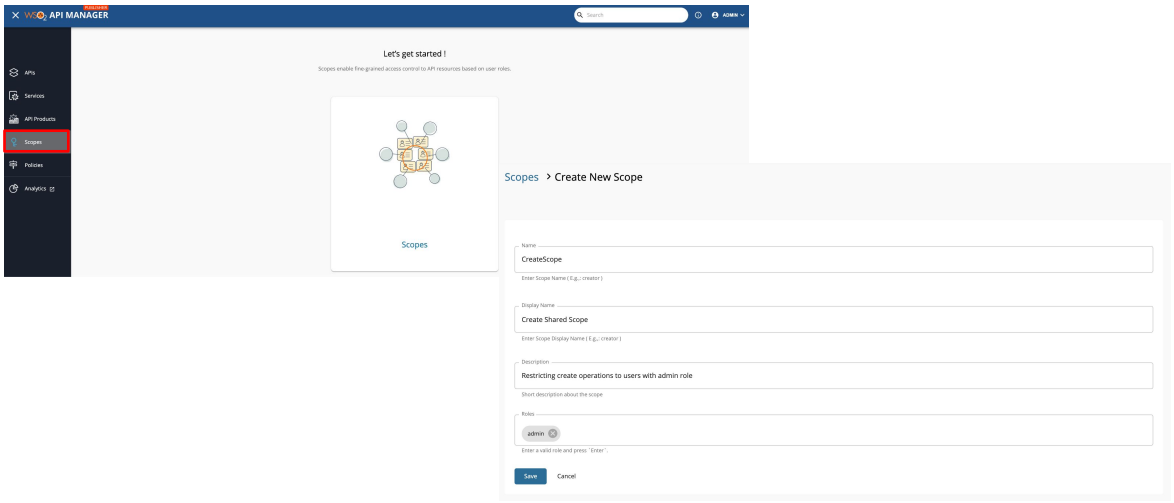
Local Scopes



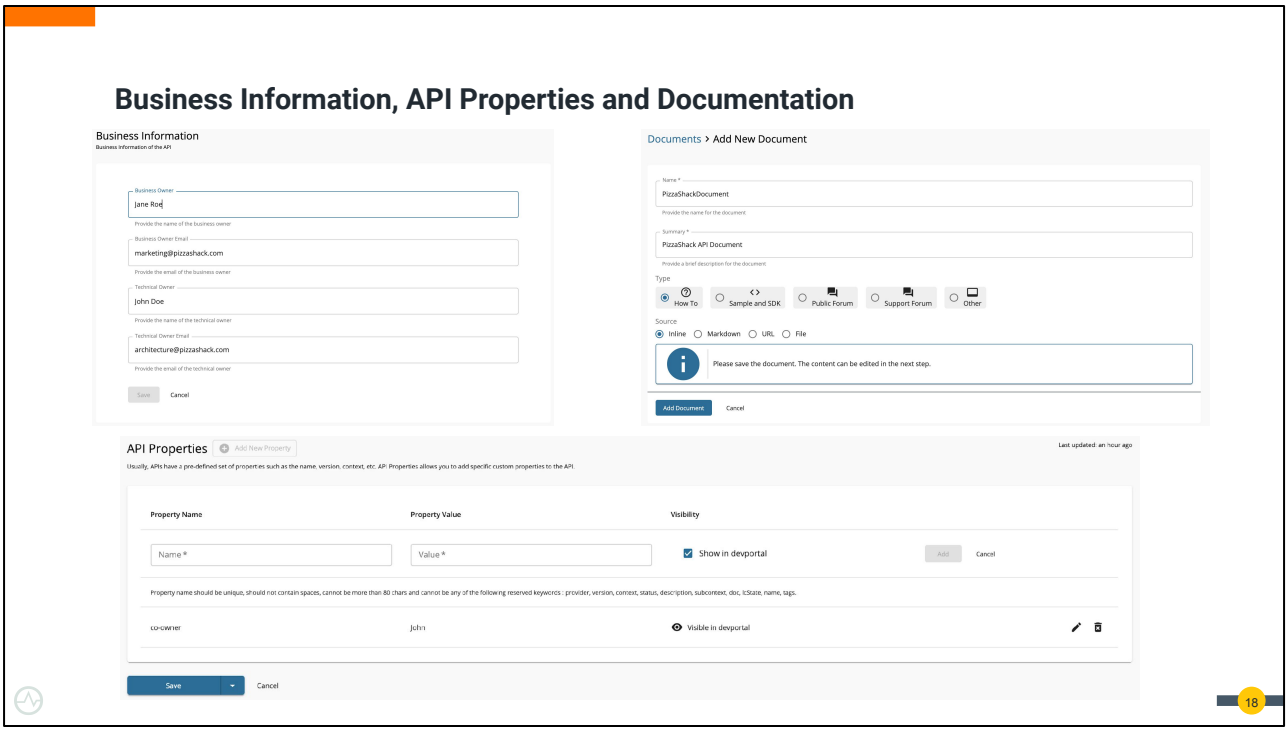
Local Scopes are a role-based access control mechanism which can be enforced to individual API resources. Local scopes are created specific to the API.

This will be discussed in detail in the API Security section.

Shared Scopes



In WSO2 API-M, an OAuth scope can be created independent of an API and be shared across multiple APIs of the same tenant.
The API-M Publisher portal provides a scope management UI to view, create, edit and delete these shared scopes.
The shared scope need to be created before API creation/update time



Define various attributes of the API.

Business Information
The business information of the API. Business owner and the technical owner details.

API Properties
Define custom parameters of the API as key value pairs. This is helpful when searching APIs

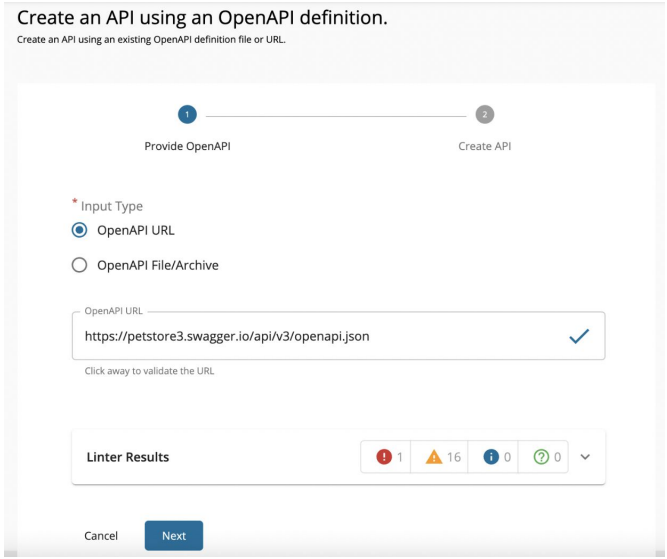
API Documentation
Add documentation for the API. It can be inline, pdf format, text format etc.

What is Swagger/OpenAPI ?

An OpenAPI file allows you to describe your entire API, including:

- Available endpoints (/users) and operations on each endpoint (GET /users, POST /users)
- Operation parameters Input and Output for each operation
- Authentication methods
- Contact information, license, terms of use and other information.
- API specifications can be written in YAML or JSON. The format is easy to learn and readable to both humans and machines.

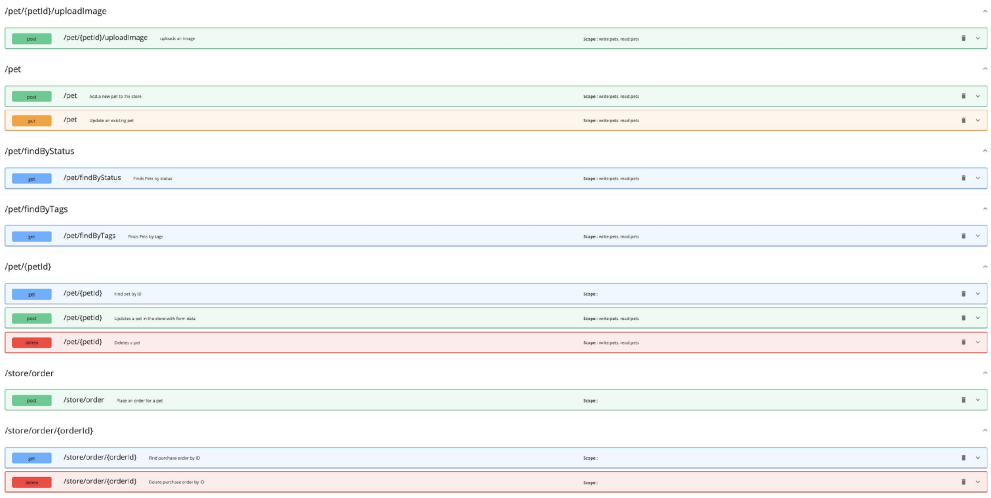
REST API from OpenAPI(Swagger) Definition



- A specification which is an API description format for REST APIs.
- Swagger 2.0 and OpenAPI 3.0
- The Specification was renamed to the OpenAPI Specification in 2015. OpenAPI 3.0 is the latest version of the specification.
- WSO2 APIM supports OpenAPI 3.0 from version 2.2.0

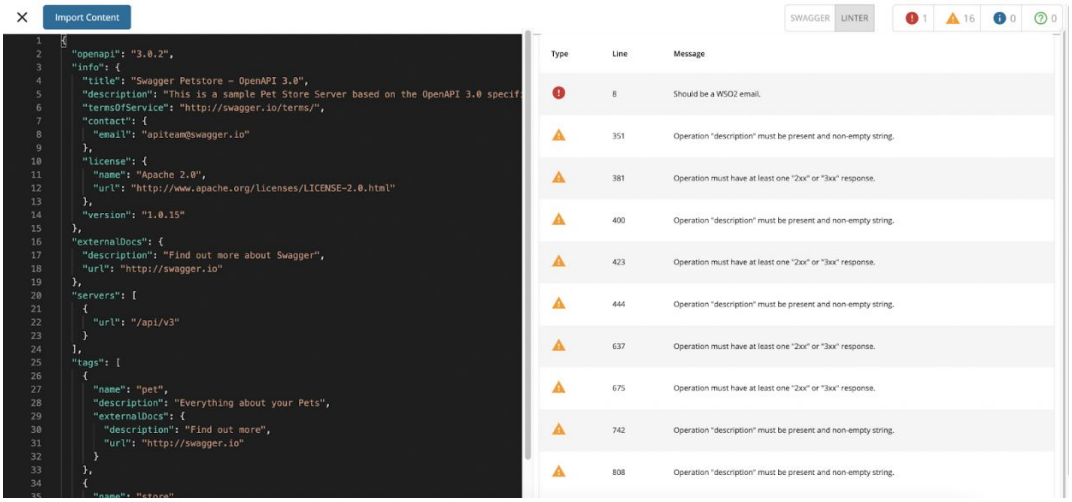
REST API from OpenAPI Definition

Resources Tab



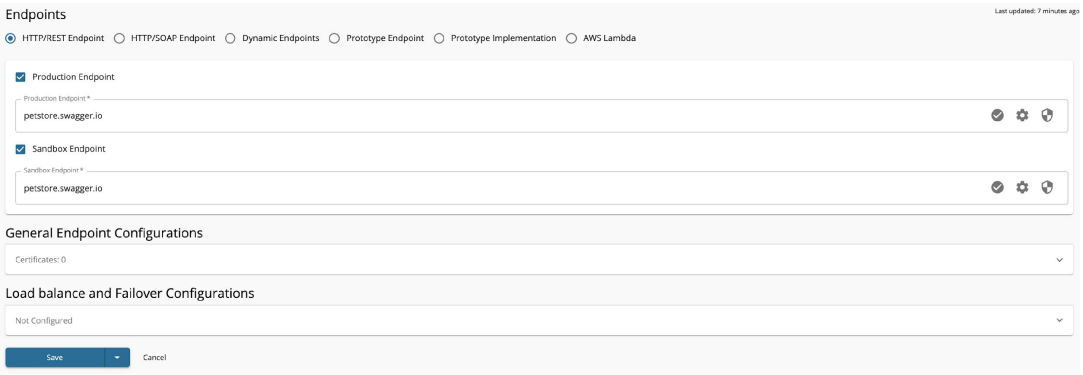
REST API from OpenAPI Definition

Resources can be edited inline



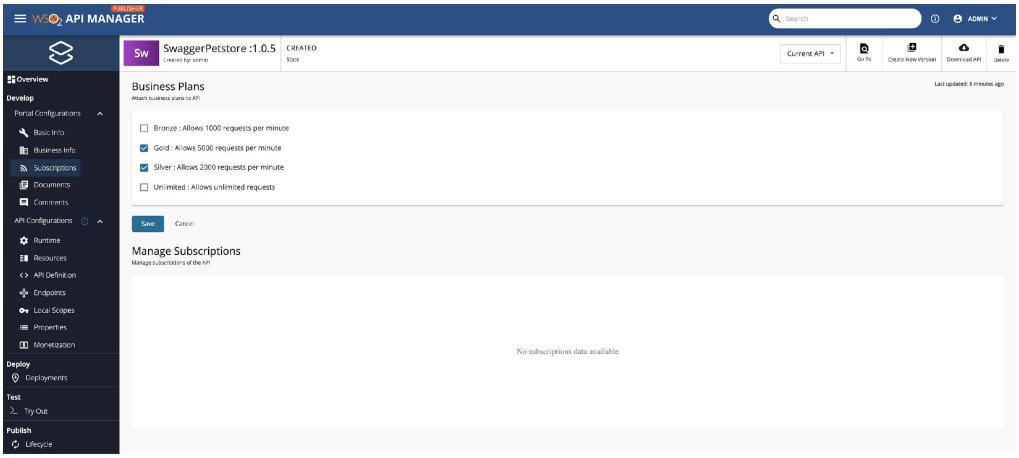
REST API from OpenAPI Definition

Configure Production and Sandbox endpoints



REST API from OpenAPI Definition

Configure Business Plans in Subscription Tab

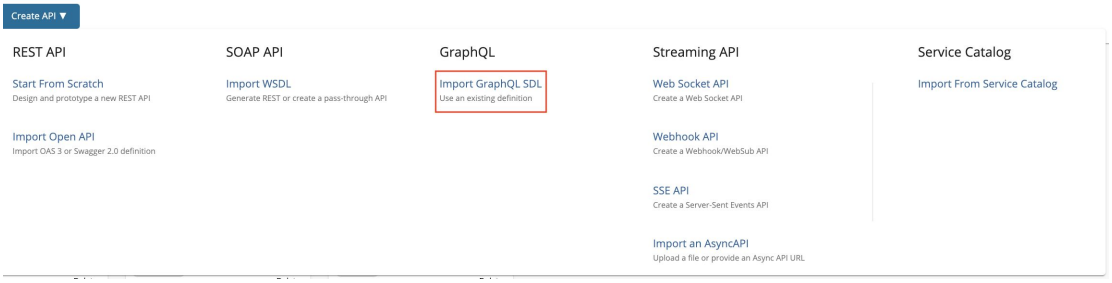


What is SOAP (Simple Object Access Protocol)?

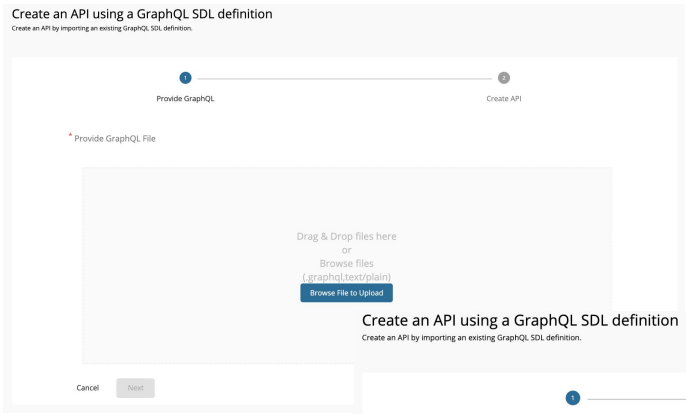
- SOAP is an XML based standard communication protocol that permits processes using different operating systems like Linux and Windows to communicate via HTTP and SOAP APIs are based on this protocol.
- REST API -> Swagger
- SOAP API -> WSDL
- The Swagger specification defines a set of files required to describe REST API.
- WSDL (Web Service Description Language), is an XML based definition language. It is used for describing the functionality of a SOAP based web service.
- SOAP APIs are deprecated in APIM 4.1.0 and it is recommended to use WSO2 Integration studio as an alternative.

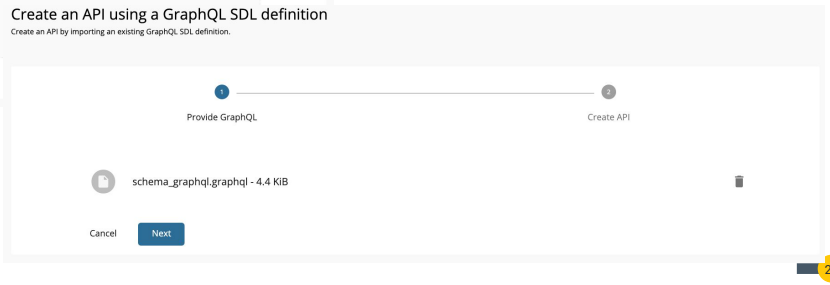
https://apim.docs.wso2.com/en/4.2.0/integrate/develop/creating-artifacts/creating-an-api/

GraphQL API from SDL



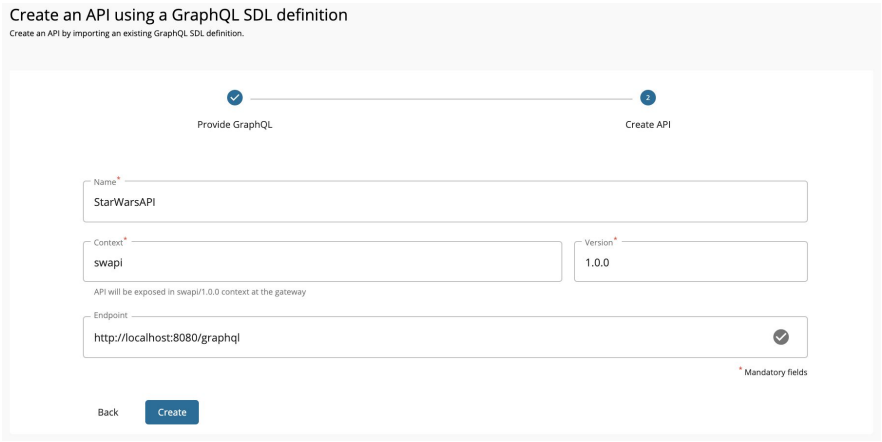
GraphQL API from SDL





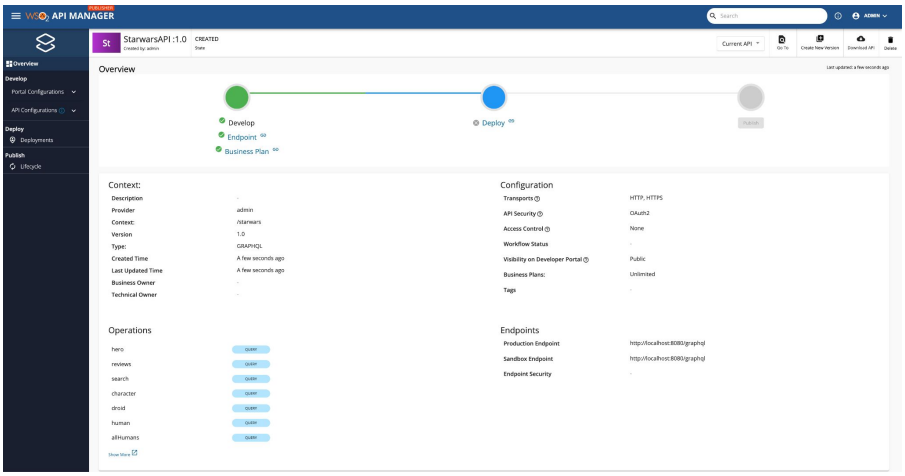
GraphQL API from SDL

Create API



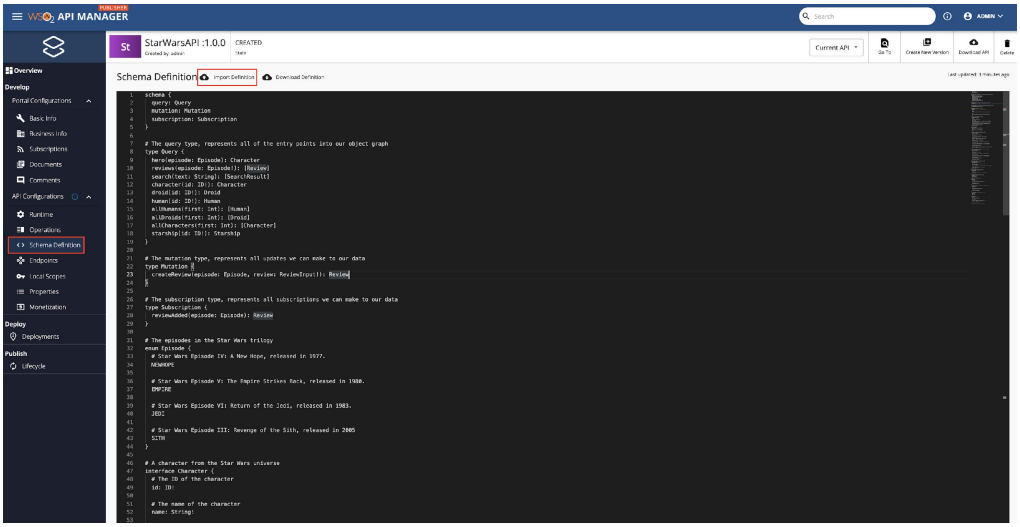
GraphQL API from SDL

API Overview



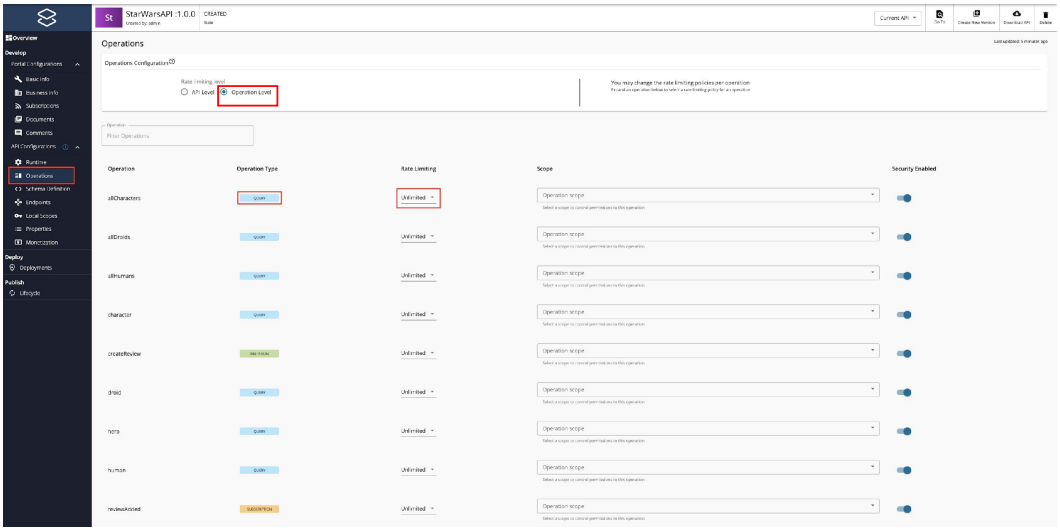
GraphQL API from SDL

Edit SDL Schema



GraphQL API from SDL

Configure GraphQL Operations



Streaming APIs

- A logical collection of related topics
- Topics represent the different channels
- Topics in Streaming APIs can be compared to the resources in REST APIs.

The actions a **topic** allows you to do are "Subscribe" and/or "Publish".

- **Subscribe** - A topic of this type allows traffic from the server (backend) to the client.
- **Publish** - A topic of this type allows traffic from the client to the server.

The Streaming Integrator component in WSO2 API Manager (WSO2 API-M) supports Streaming APIs via the following main protocols, which are compatible with HTTP.

- **WebSocket**
- **WebSub (WebHook)**
- **Server Sent Events (SSE)**

A **Streaming API** is a logical collection of related topics through which clients can publish and receive events in a well-defined format. The topics in the Streaming APIs represent the different channels. The topics in Streaming APIs can be compared to the resources in REST APIs.

The actions a **topic** allows you to do are "Subscribe" and/or "Publish".

- **Subscribe** - A topic of this type allows traffic from the server (backend) to the client.
- **Publish** - A topic of this type allows traffic from the client to the server.

The Streaming Integrator component in WSO2 API Manager (WSO2 API-M) supports Streaming APIs via the following main protocols, which are compatible with HTTP.

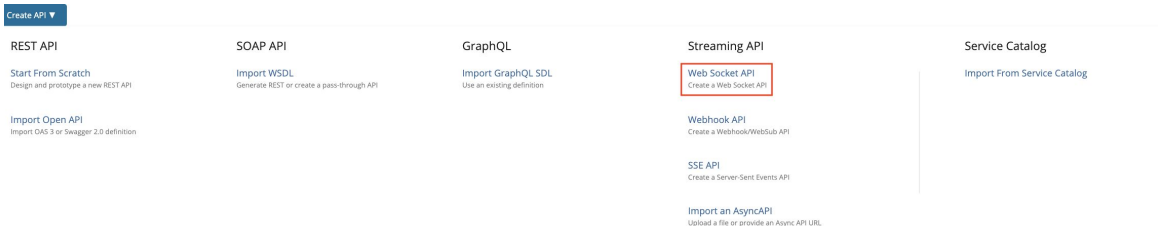
- **WebSocket**
- **WebSub (WebHook)**
- **Server Sent Events (SSE)**

a logical collection of related topics

WebSocket API

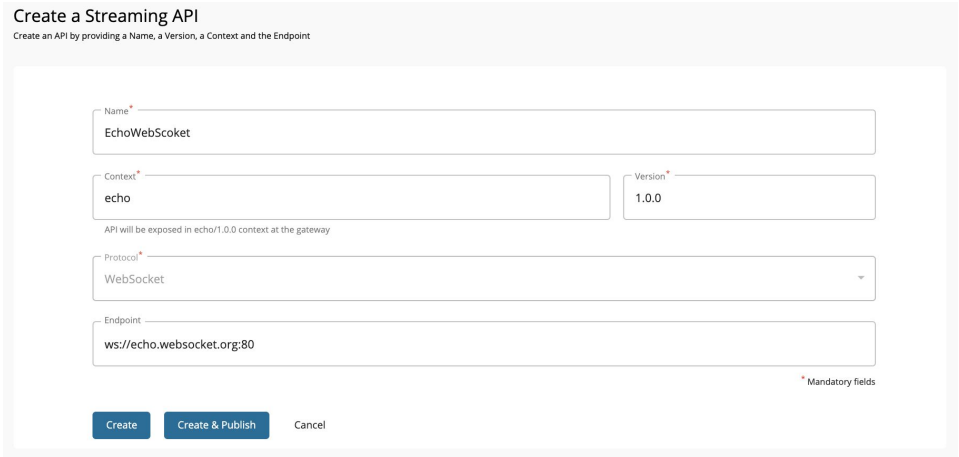
- Protocol similar to HTTP that is part of the HTML5 specification.
- Full-duplex communication between the client and the server over a single connection.
- Advantages
  - Reduce unnecessary network traffic and latency.
  - Allow streaming through proxies and firewalls while simultaneously supporting upstream and downstream communication.
  - Be backward compatible with the pre-WebSocket world by starting up as an HTTP connection before switching to WebSocket frames.

WebSocket API



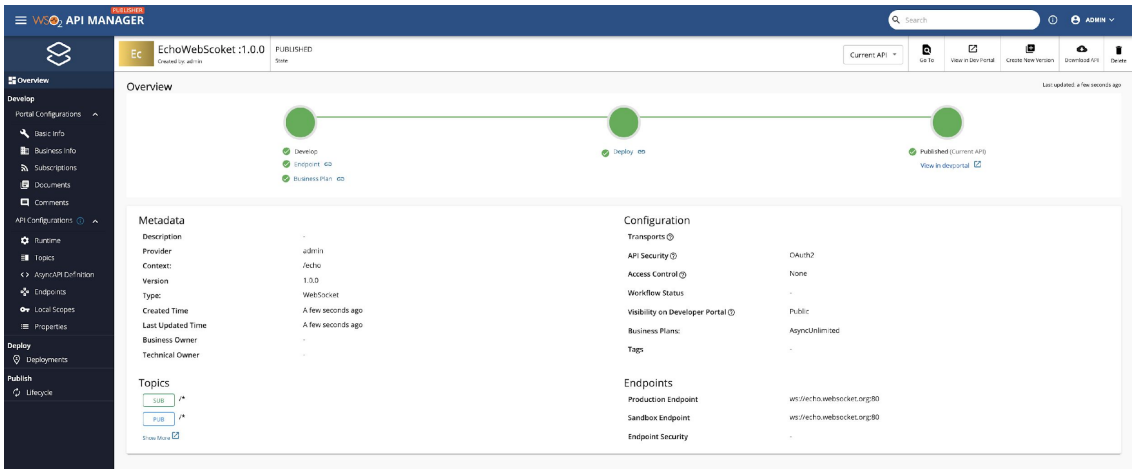
WebSocket API

Create WebSocket API



WebSocket API

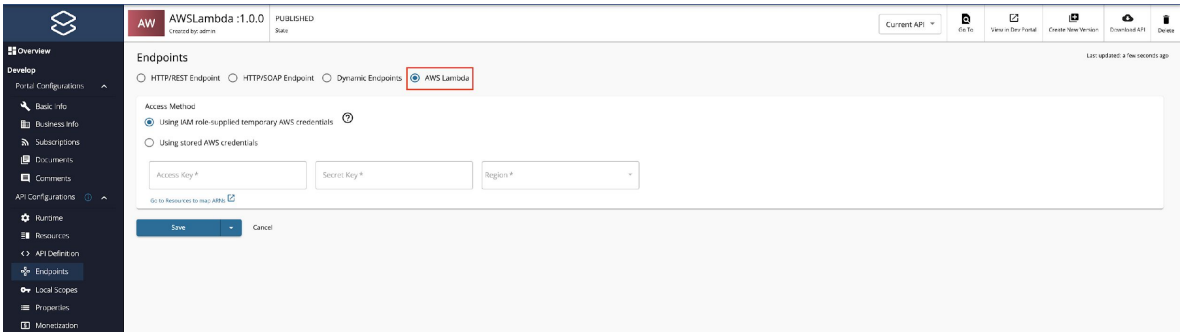
Overview WebSocket API



WebSocket API

Configuring endpoint

API with AWS Lambda Endpoint



Doc Link: [Create and Publish an AWS Lambda API]

After creating the API, select AWS Lambda as the endpoint type in Endpoints Page.

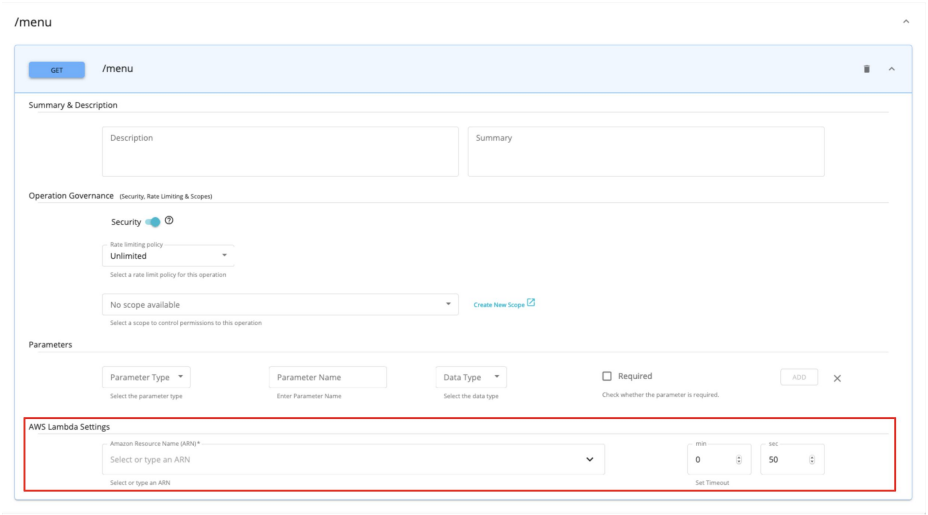
In AWS Lambda type, the credentials of the AWS API should be provided in this page.

Access Method Configuration

- **Using IAM role-supported temporary credentials** If API Manager is running on AWS EC2 instance, resolve the IAM credentials from the AWS environment.
- **Using stored AWS Credentials** Provide the Access Key, Secret and Region manually

Provide the information and save the changes.

AWS Lambda - Select Resource Level ARNs

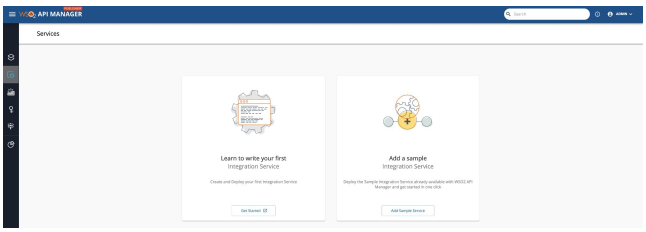


After providing the AWS credentials in Endpoints page, Amazon Resource Names can be assigned per resource.

Once configured, when invoking the API resource, the respective AWS lambda function will be invoked.

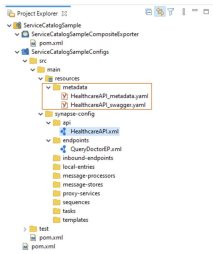
Create API Using a Service

- The service catalog in WSO2 API Manager contains services that correspond to either the Micro Integrator or the Streaming Integrator.
- The Micro Integrator services correspond to the REST API backend services
- The Streaming Integrator services correspond to the Streaming API (i.e., WebSocket, WebHook/WebSub or SSE) backend services that are managed by the respective integration layer (Micro Integrator or Streaming Integrator).



Create and Publish a REST API based on an Integration Service

1. Develop a REST API artifact using WSO2 Integration Studio
2. Update the service metadata



Parameter	Description
description	Explain the purpose of the API.
serviceUrl	This is the URL of the API when it gets deployed in the Micro Integrator. You (as the integration developer) may not know this URL during development. Therefore, you can parameterize the URL to be resolved later using environment variables. By default, the `{MI_HOST}` and `{MI_PORT}` values are parameterized with placeholders. You can configure the serviceUrl in the following ways: • Add the complete URL without parameters. For example: `http://localhost:8290/healthcare`. • Parameterize using the host and port combination. For example: `http://{MI_HOST}:{MI_PORT}/healthcare`. • Parameterize using a preconfigured URL. For example: `http://{MI_URL}/healthcare`.

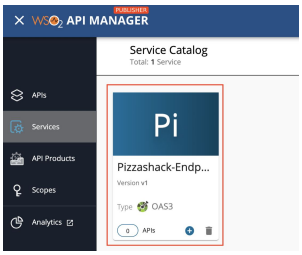
3. Configure Micro Integrator Server
4. Start the servers

```
[[service_catalog]]
apim_host = "https://localhost:9443"
enable = true
username = "admin"
password = "admin"
```

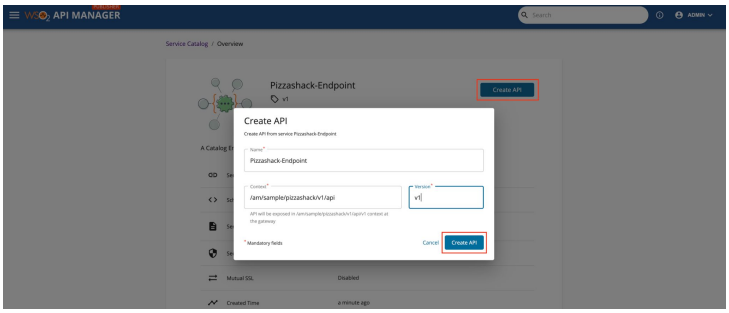
[Publish Integrations to API Manager]

Create an API Using a Service

1. Discover the service



2. Create API from the Service



[Create an API from a Service]

Managing APIs

This section covers how an created API can be managed.

Also covers, UI components related to API Management

Managing APIs

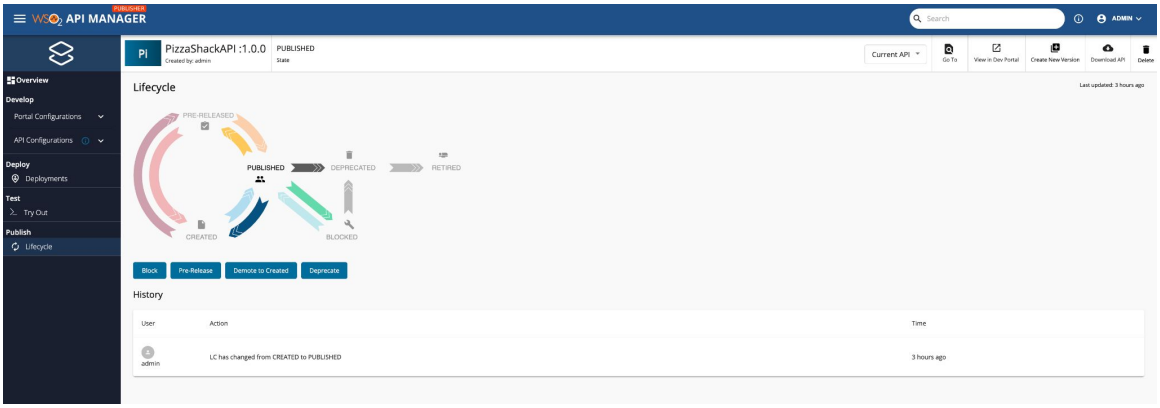
- Changing the Lifecycle State of APIs
- Deploy APIs
- API Versioning
- Managing API Policies

API Lifecycle

- States which represent the stages that an API has in the process of starting to develop an API until its retirement.
  - Created
  - Pre-Released
  - Published
  - Blocked
  - Deprecated
  - Retired

Docs Link: [API Lifecycle]

API Lifecycle



Select a created API and click on the Lifecycle menu item from the left panel.
In the lifecycle page, the diagram shows the current lifecycle state of the API and next possible lifecycle states.

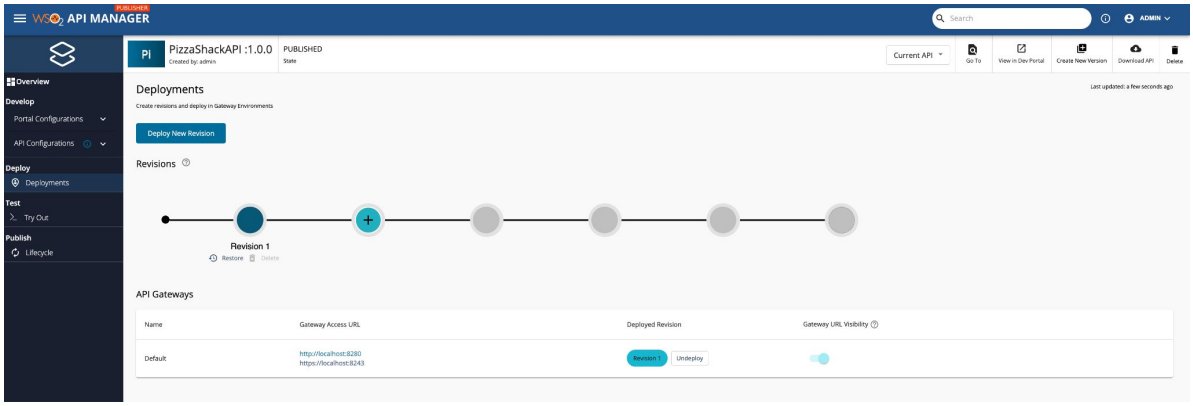
The panel in the right hand side shows the requirements to be fulfilled in order to move to the next lifecycle state.

To promote from **CREATED** state to **PUBLISHED** state, there must be a subscription policy applied as well as Endpoints should be provided.
A prototype is created for the purpose of early promotion and consumer testing. You can deploy a new API or a new version of an existing API as a prototype by using the PRE-RELEASED state. It gives subscribers an early implementation of the API.

If any required requirement is not met, the respective action is not permitted.

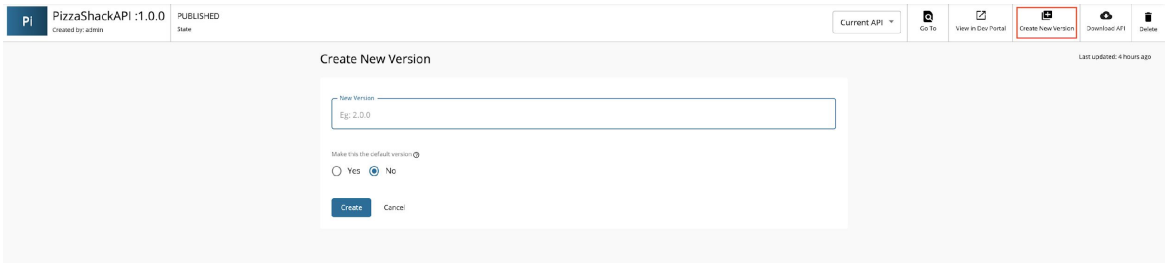
Deploying APIs

API Deploying is the process of making the API available for invocation.



API Versioning

Creating a new version of an existing API.



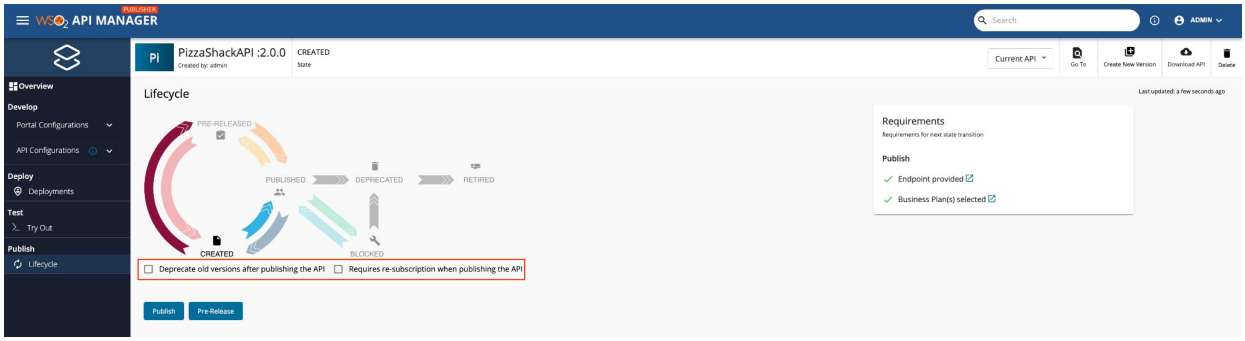
Select the API, and click “Create New Version” in the top panel.

Enter the new version.

Make this the default version: Let the API to be invoked without the version.

The new version will be in **CREATED** State. In order to publish the API, goto lifecycle page and change to **PUBLISHED**.

API Versioning



When publishing the new version following options are available.

- **Requires re-subscription when publishing the API**: If NOT CHECKED, the existing subscribers of the old version of the API will be automatically subscribed to the new version of the API. The subscribers do not need to re-subscribe. If checked, the API will be published as a new API with no subscriptions.
- **Deprecate old version after publishing the API**: Automatically deprecate the old version of the API

Managing API Policies

Enforce API Level/ resource level usage policies (Rate limiting policies) to the API.

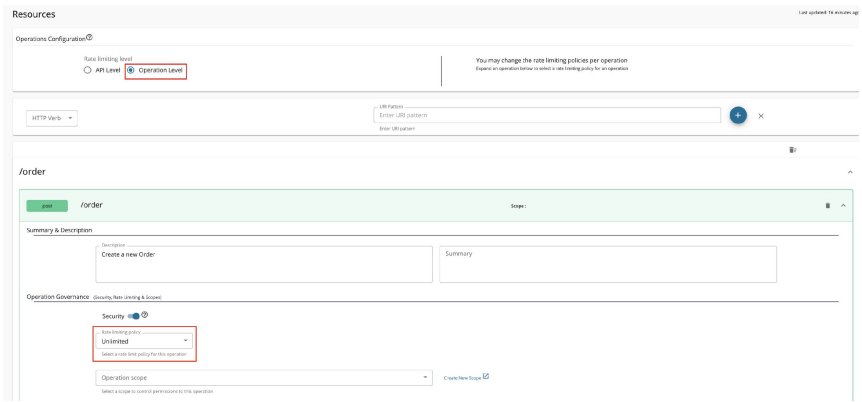
API Level Policies

Throttling policies can be added per API or per resource.

More on Throttling policies in the Throttling section

Managing API Policies

Operation Level Policies

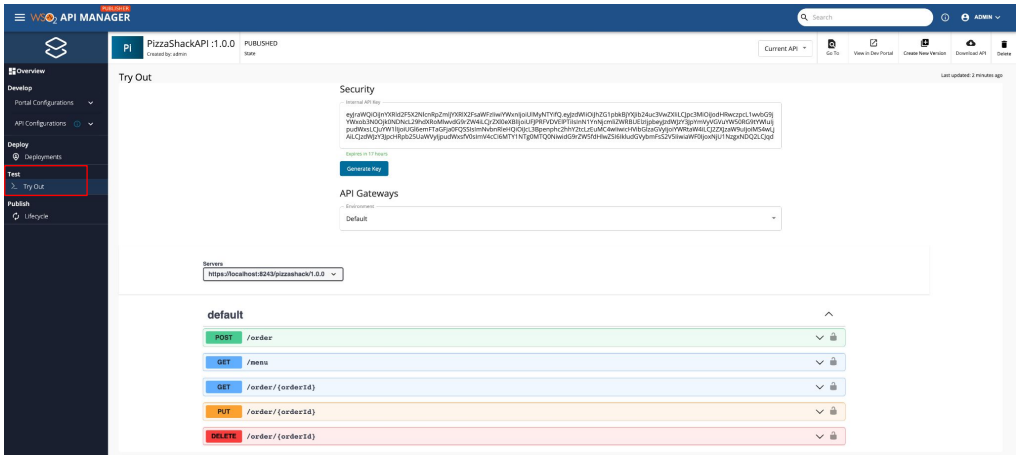


Testing APIs

This section covers how you can test a created API from the Publisher portal itself

Testing APIs

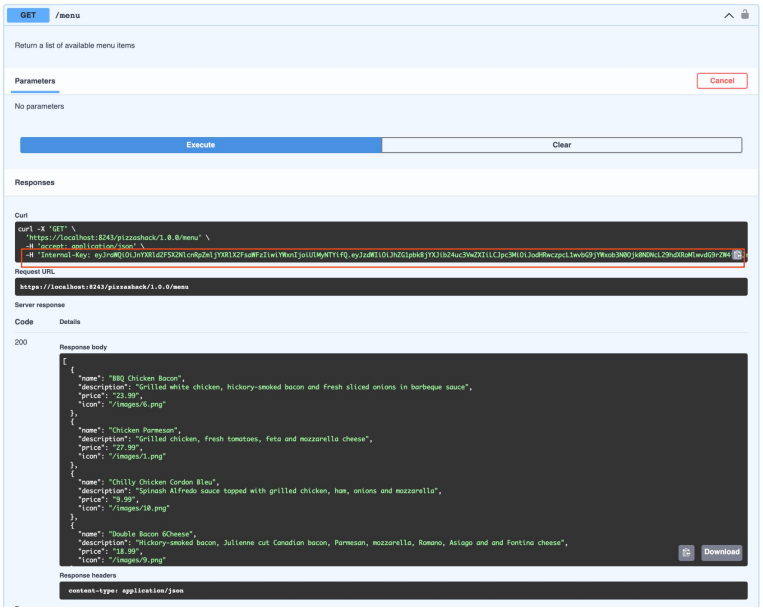
Testing APIs from the Publisher Portal (API has to be deployed first)



The Publisher Test Console provides a space to test the API before publishing and deploying an API. However, the Test Console does not provide the option to test the API in an actual Gateway with the runtime configurations. Making such changes to an API without proper testings and especially if the changes are reflected instantly in the Developer Portal and all the serving Gateways, is not a good approach for a production setup.

Testing APIs

Test the resource



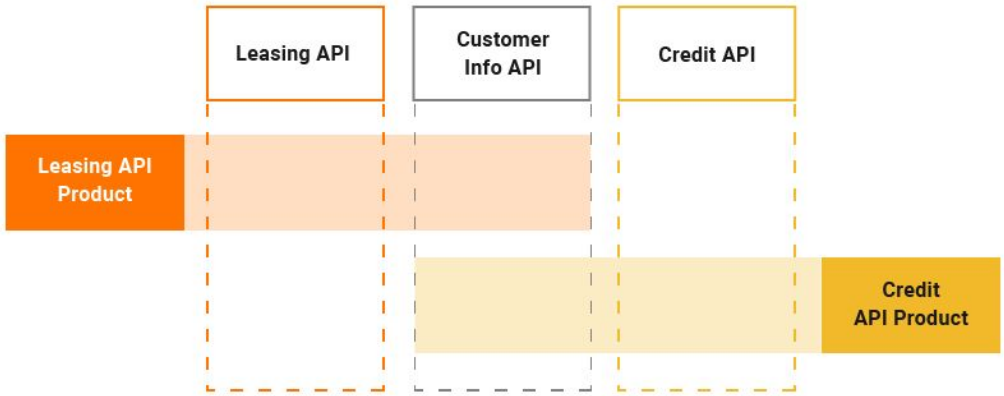
Expand the POST method and click Try it out. It will generate the test-key to invoke the API. Click Execute.

test-key token - If an API developer initiates a test, the swagger console will be automatically populated with the generated test-key. If you try the test API call through the terminal or command line, make sure you copy the generated test-key.

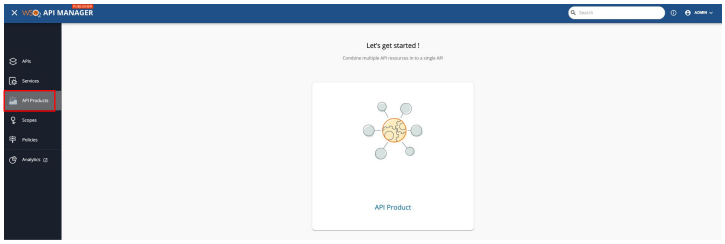
API Products

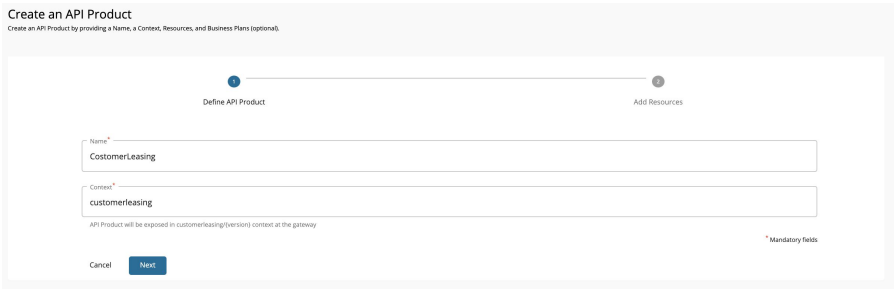
API Product

Packaging mechanism that you can use when you need to bundle a preferred set of resources from multiple APIs and expose it as a separate API interface.



Creating an API Product

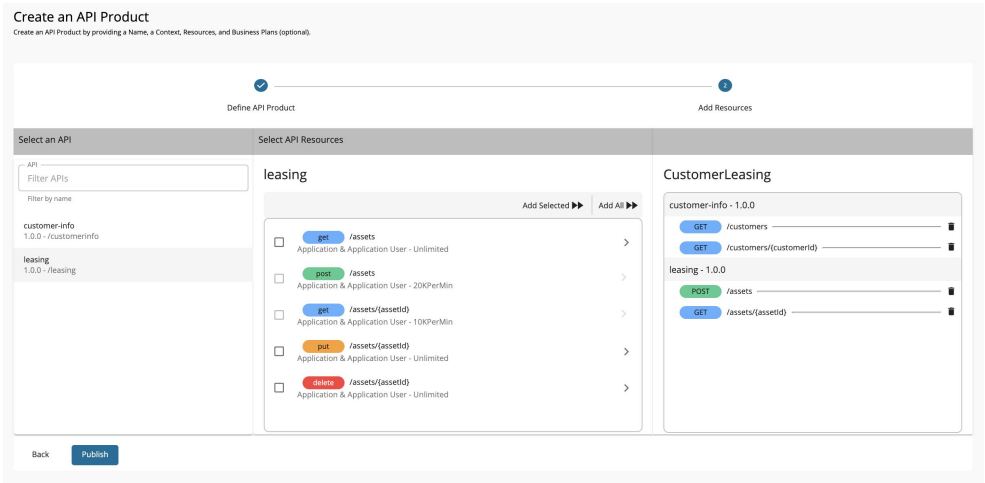




- Go to API Products by clicking on the API Products menu.
- Click Create and start creating the product by providing the name, context, and subscription tier.
- Click Next.

Publishing an API Product

Add Resources from APIs

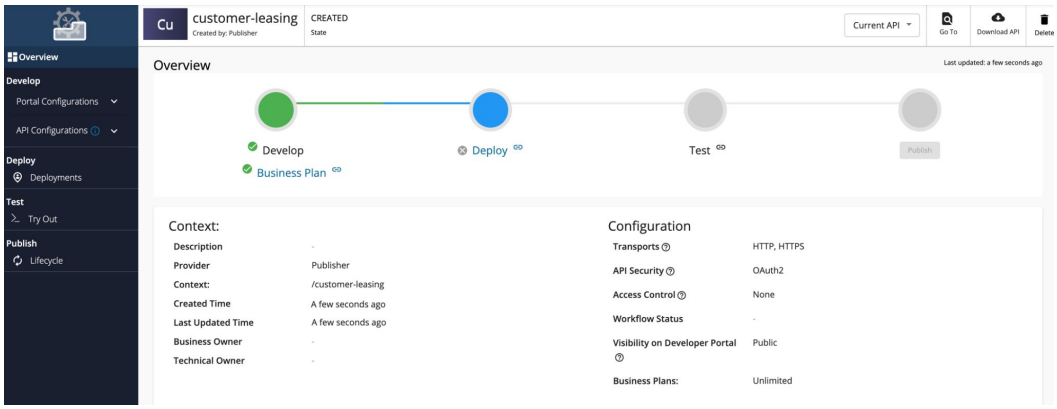


In step 2, the resources from existing APIs can be selected to create the product.

- Select the API from the left panel.
- From the resources of the selected API, check the required resources in the middle panel and click “**Add Selected**”. (It is also possible to individually add resources or add all the resources)
- When done, click **Finish**.

Publishing an API Product

Created API Product Overview



Important to note:

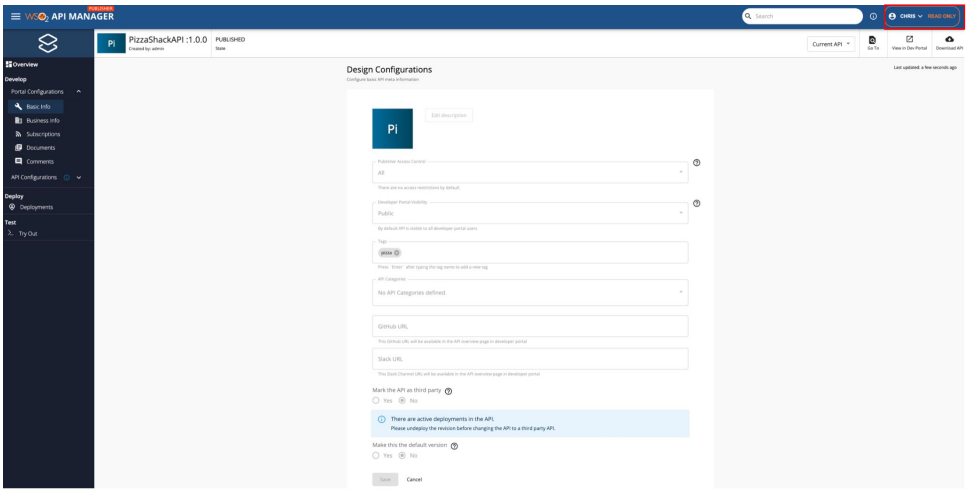
- API Products can be created from APIs which are not published (in created state)
- In order to reflect the changes done in the underlying APIs, the product should be updated manually.
- API Products are now associated with a lifecycle and the states can be changed via Lifecycle tab on left navigation menu.

Publisher Portal in Read Only Mode

- Users who have the view/read only permissions can only view the API, API Product and Service details in the Publisher Portal.
- The read only user should have the following scopes - `apim:api_view` and `apim:publisher_settings.`
- WSO2 API-M provides a pre-defined role named **internal/observer**, which is used to group all the read-only users.

https://apim.docs.wso2.com/en/4.2.0/design/api-security/authorization/publisher-portal-in-read-only-mode/#accessing-the-publisher-portal-in-read-only-mode

Publisher Portal in Read Only Mode



Let's try it out!

Creating and publishing an API Product with API Manager Publisher