



WSO2 API Manager 4.2.0 Fundamentals

[Getting Started with the API Publisher](#) WSO2 API Manager



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

Let's get started !
Choose your option to create an API

Create API ▾

REST API	SOAP API	GraphQL	Streaming API	Service Catalog
Start From Scratch Design and prototype a new REST API	Import WSDL Generate REST or create a pass-through API	Import GraphQL SDL Use an existing definition	Web Socket API Create a Web Socket API	Import From Service Catalog
Import OpenAPI Import OAS 3 or Swagger 2.0 definition			Webhook API Create a Webhook/WebSub API	
			SSE API Create a Server-Sent Events API	
			Import an AsyncAPI Upload a file or provide an Async API URL	

[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

Create API ▾

REST API Start From Scratch Design and prototype a new REST API	SOAP API Import WSDL Generate REST or create a pass-through API	GraphQL Import GraphQL SDL Use an existing definition	Streaming API Web Socket API Create a Web Socket API	Service Catalog Import From Service Catalog
Import Open API Import OAS 3 or Swagger 2.0 definition			Webhook API Create a Webhook/WebSub API	
			SSE API Create a Server-Sent Events API	
			Import an AsyncAPI Upload a file or provide an Async API URL	

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

Create an API
Create an API by providing a Name, a Version, a Context and Backend Endpoint (optional)

Name*

Context* Version*

API will be exposed in pizzashack/1.0.0 context at the gateway

Endpoint

* Mandatory fields

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

The screenshot shows the WSO2 API Manager interface for the 'PizzaShackAPI:1.0.0' API. The top navigation bar indicates the API is 'PUBLISHED'. The left sidebar has a 'Overview' tab selected, highlighted with a red box. The main content area includes:

- Overview:** A timeline showing the API's state transitions: Develop → Deploy → Test → Published (Current API). Each step has a green circular icon.
- Context:** Describes the API as a simple API for Pizza Shack online pizza delivery store. It includes fields like Provider (admin), Context (/pizzashack), Version (1.0.0), Type (HTTP), Created Time (4 minutes ago), Last Updated Time (4 minutes ago), Business Owner (Jane Doe), and Technical Owner (John Doe).
- Configuration:** Lists transports (HTTP, HTTPS), API Security (OAuth2), Access Control (None), Workflow Status, Visibility on Developer Portal (Public), Business Plans (Unlimited), and Tags (pizza).
- Resources:** Lists three resources: /order (with POST method), /menu (with GET method), and /order/{orderId} (with GET, PUT, and DELETE methods). Each resource entry includes a 'Show More' link.
- Endpoints:** Lists two endpoints: Production Endpoint (<https://localhost:9443/em/sample/pizzashack/v1/api/>) and Sandbox Endpoint (<https://localhost:9443/em/sample/pizzashack/v1/api/>). It also includes Endpoint Security information.

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

The screenshot shows the WSO2 API Manager interface. On the left, there's a sidebar with sections like Overview, Develop, Test, and Deploy. Under Develop, 'Portal Configurations' is expanded, and 'Basic info' is selected, indicated by a red box. The main content area is titled 'Design Configurations' and contains fields for Publisher Access Control (set to All), Developer Portal Visibility (set to Public), Tags (containing 'pizza'), GitHub URL, and Slack URL. A note says 'This slack channel URL will be available in the API overview page in developer portal.' Below these fields, there are two sections: 'Mark the API as third party' (radio button set to 'No') and 'Make this the default version' (radio button set to 'No'). A blue callout box highlights a note: 'There are active deployments in the API. Please undeploy the version before changing the API to a third party API.' At the bottom are 'Save' and 'Cancel' buttons.

Design Configuration includes,

- Publisher Access control in the Publisher portal
- Developer Portal Visibility in the Developer Portal
- Tags
- API Categories
- GitHub URL
- Slack URL
- Mark this API as third party gateway
- Make the API the default version

To which roles, the API should be visible

To which roles the API should be visible

Add tags to the API.
Set API categories

API will not be proxied through the gateway

API Runtime Configurations

The screenshot shows the WSO2 API Manager interface. On the left, the sidebar has sections for Overview, Develop (Portal Configurations, API Configurations, Resources, API Definition, Endpoints, Local Scopes, Policies, Properties), Deploy (Deployments), and Test (Try Out). The 'API Configurations' section is expanded, and the 'Runtime' option is selected and highlighted with a red box. The main content area is titled 'Runtime Configurations'. It is divided into 'Request' and 'Response' sections on the left and a 'Backend' section on the right. The 'Request' section contains dropdowns for 'Transport Level Security' and 'Application Level Security', and toggle switches for 'CORS Configuration' and 'Schema Validation'. The 'Response' section contains a toggle switch for 'Response Caching'. The 'Backend' section includes 'Backend Throughput' (with options for 'Unlimited' or 'Specify'), 'Endpoints' (listing 'Production' and 'Sandbox' URLs), and a 'Edit API Endpoints' button. At the bottom are 'Save' and 'Cancel' buttons.

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.

The screenshot shows the WSO2 API Manager interface under the 'API Resources' section. The sidebar on the left is collapsed. The main header shows 'PUBLISHED' status for 'PizzaShackAPI :1.0.0'. The left sidebar has several sections: Overview, Develop, API Configurations (with 'Runtime' and 'Resources' selected), Endpoints, Local Scopes, Policies, Properties, Deploy (with 'Deployments' listed), and Test (with 'Try Out' option). The main content area is titled 'Operations Configuration' and includes a 'Rate limiting level' section with 'API Level' and 'Operation Level' options. Below this are three resource sections: '/order' (POST method), '/menu' (GET method), and '/order/{orderId}' (with GET, PUT, and DELETE methods). Each resource section has a 'Save' and 'Reset' button at the bottom.

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.

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

The screenshot shows the WSO2 API Manager interface. On the left, there's a sidebar with navigation links: Overview, Develop, API Configurations, Deploy, Test, and Publish. Under Develop, the 'Subscriptions' link is highlighted with a red box. The main content area shows details for the 'PizzaShackAPI :1.0.0' which is 'PUBLISHED'. It has a status of 'State'. Below this, the 'Business Plans' section lists tiers: Bronze (1000 requests/min), Gold (5000 requests/min), Silver (2000 requests/min), and Unlimited (unlimited requests). The 'Unlimited' option is checked. A 'Save' button is present. Below this is the 'Manage Subscriptions' section, which shows one subscriber named 'subscriber' associated with the application 'TestApp', tier 'Unlimited', and status 'UNBLOCKED'. Actions for this subscriber include 'Block Production Only', 'Block All', and 'Unlock'. A 'View Invoice' button is also available. The bottom right of the page shows 'Rows per page: 5' and '1-1 of 1'.

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

The screenshot shows the WSO2 API Manager interface. On the left, the sidebar includes sections for Overview, Develop, API Configurations, Runtime, Resources, API Definition (which is selected), Endpoints, Policies, Properties, Monetization, Deploy, Deployments, Test, Try Out, Publish, and Lifecycle. The main content area is titled "API Definition" for "PizzaShackAPI 1.0.0 PUBLISHED". It shows the following details:

- API Configuration:** pizzaShackAPI
- Descriptions:** A successful API for Pizza Shack online pizza delivery. (This)
- Contact:** Name: Pizza Shack, URL: http://www.pizzashack.com, Email: architecture@pizzashack.com
- Licenses:** Apache 2.0, URL: http://www.apache.org/licenses/LICENSE-2.0.html, Version: 1.0.0
- Paths:** /order/{orderId}
- Post:**
 - Description: Create a new order.
 - Request body: { "order": "Pepsi", "quantity": 1 }
 - Created: Successful response with the newly created object (note body.location header contains URL of newly created entity).
 - Location: The URL of the newly created resource.
 - Styles: simple
 - Explodes: false
 - Content: string
 - Type: STRING
 - Consumes: application/json
 - Descriptions: The content type of the body.
 - Style: simple
 - Produces: application/json
 - Schemas:
 - content: string
 - schema: draft: /components/schemas/Order
 - examples: Bad Request: Invalid request or validation error.

Operations:

- default**
 - GET /order**
 - PUT /order/{orderId}**
 - PATCH /order/{orderId}**
 - DELETE /order/{orderId}**

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

WSO2 API MANAGER

PUBLISHED

PI PizzaShackAPI :1.0.0 PUBLISHED State

Search Current API Go To View in Dev Portal Download API

Overview Develop Deploy Test Publish

Portal Configurations API Configurations Deploy Deployments Try Out Lifecycle

Deployments Create revisions and deploy in Gateway Environments Deploy New Revision

Revisions Last updated: 2 hours ago

Revision 1 + Restore Delete

API Gateways

Name	Gateway Access URL	Deployed Revision	Gateway URL Visibility
Default	http://localhost:8280 https://localhost:8243	Revision 1 Undeploy	On

The screenshot shows the WSO2 API Manager interface. At the top, it displays the API 'PizzaShackAPI :1.0.0' in a 'PUBLISHED' state. The left sidebar has sections for Overview, Develop, Deploy (with 'Deployments' highlighted), Test, and Publish. Under Deploy, there are options for Portal Configurations, API Configurations, and Lifecycle. The main content area shows 'Deployments' and 'API Gateways'. In the Deployments section, there's a timeline showing 'Revision 1' (the first node is solid blue, others are grey). Below the timeline are 'Restore' and 'Delete' buttons. In the API Gateways section, there's a table with one row for 'Default'. The table columns are 'Name', 'Gateway Access URL', 'Deployed Revision', and 'Gateway URL Visibility'. The 'Gateway Access URL' row contains 'http://localhost:8280' and 'https://localhost:8243'. The 'Deployed Revision' row contains a button labeled 'Revision 1 Undeploy'. The 'Gateway URL Visibility' row has a toggle switch set to 'On'.

Local Scopes

The screenshot shows the WSO2 API Manager interface for the 'PizzaShackAPI:1.0.0' API. The left sidebar has a 'Local Scopes' option highlighted with a red box. The main content area shows a form for creating a new scope:

- Name: OrderPizza
- Display Name: Order Pizza
- Description: Order Pizza
- Roles: admin

At the bottom are 'Save' and 'Cancel' buttons.

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

The screenshot shows the WSO2 API Manager Publisher portal. On the left, a sidebar menu includes 'APIs', 'Services', 'API Products', 'Scopes' (which is highlighted with a red box), 'Policies', and 'Analytics'. The main content area has a title 'Let's get started!' and a sub-instruction 'Scopes enable fine-grained access control to API resources based on user roles.' Below this is a central panel titled 'Scopes > Create New Scope'. It features a circular icon with various icons representing different scope types. A form on the right is titled 'CreateScope'. It contains fields for 'Name' (set to 'CreateScope'), 'Display Name' (set to 'Create Shared Scope'), 'Description' (set to 'Restricting create operations to users with admin role'), and 'Roles' (set to 'admin'). A note below the roles field says 'Enter a valid role and press "Enter".' At the bottom of the form are 'Save' and 'Cancel' buttons.

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

Business Information, API Properties and Documentation

The screenshot displays three main sections of the API management interface:

- Business Information:** A form for entering business owner details. Fields include:
 - Business Owner: Jane Doe
 - Business Owner Email: marketing@pizzashack.com
 - Technical Owner: John Doe
 - Technical Owner Email: architecture@pizzashack.com
- Documents > Add New Document:** A form for creating a new document. Fields include:
 - Name: PizzaShackDocument
 - Summary: PizzaShack API Document
 - Type: How To (selected)
 - Source: Inline (selected)
 - Description: Please save the document. The content can be edited in the next step.
- API Properties:** A table for managing API properties. It shows one row:

Property Name	Property Value	Visibility
Name *	Value *	<input checked="" type="checkbox"/> Show in devportal

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

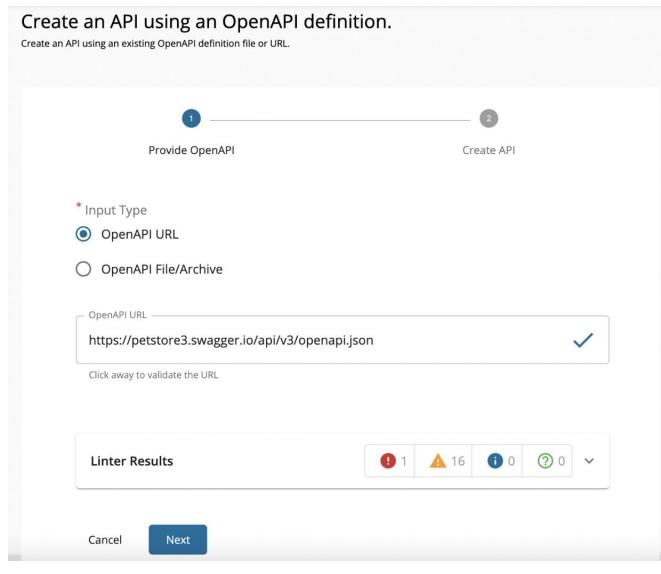
Add documentation for the API. It can be inline, pdf format, text format etc.

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

/pet/{petId}/uploadImage			
	/pet/{petId}/uploadImage	uploads an image	Scope: write:pets, read:pets
<hr/>			
	/pet	Add a new pet to the store	Scope: write:pets, read:pets
	/pet	Updates an existing pet	Scope: write:pets, read:pets
<hr/>			
/pet/findByStatus			
	/pet/findByStatus	Find Pets by status	Scope: write:pets, read:pets
<hr/>			
/pet/findByTags			
	/pet/findByTags	Find Pets by tags	Scope: write:pets, read:pets
<hr/>			
/pet/{petId}			
	/pet/{petId}	Find pet by ID	Scope:
	/pet/{petId}	Updates a pet in the store with form data	Scope: write:pets, read:pets
	/pet/{petId}	Deletes a pet	Scope: write:pets, read:pets
<hr/>			
/store/order			
	/store/order	Place an order for a pet	Scope:
<hr/>			
/store/order/{orderId}			
	/store/order/{orderId}	Find purchase order by ID	Scope:
	/store/order/{orderId}	Deletes purchase order by ID	Scope:

REST API from OpenAPI Definition

Resources can be edited inline

```
X Import Content
1
2   "openapi": "3.0.2",
3   "info": {
4     "title": "Swagger Petstore - OpenAPI 3.0",
5     "description": "This is a simple Pet Store Server based on the OpenAPI 3.0 specification",
6     "termsOfService": "http://swagger.io/terms/",
7     "contact": {
8       "email": "apiteam@swagger.io"
9     },
10    "license": {
11      "name": "Apache 2.0",
12      "url": "http://www.apache.org/licenses/LICENSE-2.0.html"
13    },
14    "version": "1.0.15"
15  },
16  "externalDocs": {
17    "description": "Find out more about Swagger",
18    "url": "http://swagger.io"
19  },
20  "servers": [
21    {
22      "url": "/api/v3"
23    }
24  ],
25  "tags": [
26    {
27      "name": "pet",
28      "description": "Everything about your Pets",
29      "externalDocs": {
30        "description": "Find out more",
31        "url": "http://swagger.io"
32      }
33    },
34    {
35      "name": "store"
36    }
37  ]
38}
```

Type	Line	Message
!	8	Should be a WSO2 email.
!	351	Operation "description" must be present and non-empty string.
!	381	Operation must have at least one "2xx" or "3xx" response.
!	400	Operation "description" must be present and non-empty string.
!	423	Operation must have at least one "2xx" or "3xx" response.
!	444	Operation "description" must be present and non-empty string.
!	637	Operation must have at least one "2xx" or "3xx" response.
!	675	Operation must have at least one "2xx" or "3xx" response.
!	742	Operation "description" must be present and non-empty string.
!	808	Operation "description" must be present and non-empty string.

REST API from OpenAPI Definition

Configure Production and Sandbox endpoints

Last updated: 7 minutes ago

Endpoints

HTTP/REST Endpoint HTTP/SOAP Endpoint Dynamic Endpoints Prototype Endpoint Prototype Implementation AWS Lambda

Production Endpoint
Production Endpoint *
petstore.swagger.io (checkmark) (gear) (key)

Sandbox Endpoint
Sandbox Endpoint *
petstore.swagger.io (checkmark) (gear) (key)

General Endpoint Configurations

Certificates: 0 (dropdown)

Load balance and Failover Configurations

Not Configured (dropdown)

Save Cancel

REST API from OpenAPI Definition

Configure Business Plans in Subscription Tab

The screenshot shows the WSO2 API Manager interface. On the left, there's a sidebar with navigation links like Overview, Develop, API Configurations, Deploy, Test, Publish, and Lifecycle. The main area is titled "SwaggerPetstore :1.0.5" and shows a "Business Plans" section. It lists four plan options: Bronze (1000 requests/min), Gold (5000 requests/min, checked), Silver (2000 requests/min), and Unlimited (unlimited requests). Below this is a "Manage Subscriptions" section which states "No subscription data available". At the top right, there are buttons for Current API, Go To, Create New Version, Download API, and Delete. A search bar and an ADMIN dropdown are also at the top.

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.

 This feature is deprecated. Please use the WSO2 Integration Studio as an alternative
Integration Studio Documentation:<https://apim.docs.wso2.com/en/4.1.0/integrate/develop/creating-artifacts/creating-an-api/>



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

GraphQL API from SDL

Create API ▾

REST API

Start From Scratch
Design and prototype a new REST API

SOAP API

Import WSDL
Generate REST or create a pass-through API

GraphQL

Import GraphQL SDL
Use an existing definition

Streaming API

Web Socket API
Create a Web Socket API

Service Catalog

Import From Service Catalog

Import Open API

Import OAS 3 or Swagger 2.0 definition

Webhook API

Create a Webhook/WebSub API

SSE API

Create a Server-Sent Events API

Import an AsyncAPI

Upload a file or provide an Async API URL



GraphQL API from SDL

Create an API using a GraphQL SDL definition
Create an API by importing an existing GraphQL SDL definition.

Provide GraphQL

* Provide GraphQL File

Drag & Drop files here
or
Browse files
(graphql/text/plain)
Browse File to Upload

Create an API using a GraphQL SDL definition

Create an API by importing an existing GraphQL SDL definition.

Cancel

Next

Provide GraphQL

Create API



schema_graphql.graphql - 4.4 KiB

Cancel

Next

27

GraphQL API from SDL

Create API

Create an API using a GraphQL SDL definition

Create an API by importing an existing GraphQL SDL definition.

Provide GraphQL

Create API

Name*

StarWarsAPI

Context*

swapi

Version*

1.0.0

API will be exposed in swapi/1.0.0 context at the gateway

Endpoint

http://localhost:8080/graphql

* Mandatory fields

Back

Create



28

GraphQL API from SDL

API Overview

The screenshot shows the WSO2 API Manager interface for the StarwarsAPI-1.0 version. The top navigation bar includes 'WSO2 API MANAGER', a search bar, and 'ADMIN' dropdown. The main header displays 'StarwarsAPI-1.0' and 'CREATED'. On the left, a sidebar menu lists 'Overview', 'Desktop', 'Portal Configurations', 'API Configurations', 'Deploy', 'Deployments', 'Push', and 'Lifecycle'. The central 'Overview' section features a network diagram with nodes 'Develop' (green), 'Deploy' (blue), and 'Business Plan' (grey). Below the diagram, the 'Context' section provides details like Provider (admin), Context (starwars), Version (1.0), Type (GraphQL), and Created Time (A few seconds ago). The 'Configuration' section lists transports (HTTP, HTTPS), API Security (OAuth2), Access Control (None), Workflow Status (Not Started), Visibility on Developer Portal (Public), Business Plans (Unlimited), and Tags. The 'Operations' section lists various GraphQL operations: hero, reviews, search, character, droid, human, and allHumans, each with a 'Query' button. The 'Endpoints' section shows Production Endpoint (http://localhost:8080/graphql) and Sandbox Endpoint (http://localhost:8080/graphql). A note at the bottom right says 'Last updated a few seconds ago'.

GraphQL API from SDL

Edit SDL Schema

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

- Left Sidebar:** Shows the navigation menu with sections like Overview, Develop, Configuration, Runtime, Operations, and Deploy.
- Current API:** StarWarsAPI:1.0.0 (CREATED)
- Schema Definition:** A code editor containing the GraphQL schema definition for the Star Wars API.
- Toolbar:** Includes buttons for Import Definition, Download Definition, Current API, API Version, Create New API, Delete API, and Logout.

```
schema {  
    query: Query  
    mutation: Mutation  
    subscription: Subscription  
}  
  
# The query type, represents all of the entry points into our object graph  
query {  
    hero(id: ID!): Character  
    reviews(episodes: [ID]): [Review]  
    search(term: String): [Result]  
    character(id: ID!): Character  
    planet(id: ID!): Planet  
    human(id: ID!): Human  
    alien(id: ID!): Alien  
    species(id: ID!): Species  
    vehicle(id: ID!): Vehicle  
    starship(id: ID!): Starship  
}  
  
# The mutation type, represents all updates we can make to our data  
type Mutation {  
    reviewMovieOnEpisode(episode: Episode!, review: ReviewInput!): Review  
}  
  
# The subscription type, represents all subscriptions we can make to our data  
type Subscription {  
    newReleasedEpisodes: Episode  
}  
  
# The episodes in the Star Wars trilogy  
enum Episode {  
    EP1  
    EP2  
    EP3  
}  
# Star Wars Episode IV: A New Hope, released in 1977.  
# NAME  
#  
# Star Wars Episode V: The Empire Strikes Back, released in 1980.  
# EPISODE  
#  
# Star Wars Episode VII: Return of the Jedi, released in 1983.  
#  
# Star Wars Episode III: Revenge of the Sith, released in 2005  
# SITH  
#  
# A character from the Star Wars universe  
#  
# The character  
# The ID of the character  
# id: ID!  
#  
# The name of the character  
# name: String!
```

GraphQL API from SDL

Configure GraphQL Operations

The screenshot shows the configuration interface for the StarWarsAPI version 1.0.0. The left sidebar has sections for Overview, DevTools, and Deploy. Under DevTools, 'Runtime' is selected, which includes 'Observations', 'XSchema Definition', 'Endpoints', 'Local Issues', 'Properties', and 'Migration'. Under Deploy, there are sections for 'Deployments', 'Available', and 'Lifecycle'. The main content area is titled 'Operations' and shows a table of operations. The first row, 'Rate limiting level', has two options: 'At Level' and 'Operation Level', with 'Operation Level' being selected and highlighted with a red box. The table columns are 'Operation', 'Operation Type', 'Rate Limiting', 'Scope', and 'Security Enabled'. The operations listed are: allCharacters, allDroids, allHumans, character, createDroid, Droid, hero, human, and undeleted. Most operations have their operation type set to 'Query', while 'createDroid' is set to 'Mutation'. All operations have 'Unlimited' rate limiting. The 'Scope' dropdown for each operation is set to 'Operation scope' with the note 'Select a scope or create new scopes for this operation'. The 'Security Enabled' switch is turned on for all operations.

Operation	Operation Type	Rate Limiting	Scope	Security Enabled
allCharacters	Query	Unlimited	Operation scope Select a scope or create new scopes for this operation	Enabled
allDroids	Query	Unlimited	Operation scope Select a scope or create new scopes for this operation	Enabled
allHumans	Query	Unlimited	Operation scope Select a scope or create new scopes for this operation	Enabled
character	Query	Unlimited	Operation scope Select a scope or create new scopes for this operation	Enabled
createDroid	Mutation	Unlimited	Operation scope Select a scope or create new scopes for this operation	Enabled
Droid	Query	Unlimited	Operation scope Select a scope or create new scopes for this operation	Enabled
hero	Query	Unlimited	Operation scope Select a scope or create new scopes for this operation	Enabled
human	Query	Unlimited	Operation scope Select a scope or create new scopes for this operation	Enabled
undeleted	Subscription	Unlimited	Operation scope Select a scope or create new scopes for this operation	Enabled

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

Create API ▾

REST API

[Start From Scratch](#)
Design and prototype a new REST API

[Import Open API](#)
Import OAS 3 or Swagger 2.0 definition

SOAP API

[Import WSDL](#)
Generate REST or create a pass-through API

GraphQL

[Import GraphQL SDL](#)
Use an existing definition

Streaming API

[Web Socket API](#)
Create a Web Socket API

[Webhook API](#)
Create a Webhook/WebSub API

[SSE API](#)
Create a Server-Sent Events API

Service Catalog

[Import From Service Catalog](#)

[Import an AsyncAPI](#)
Upload a file or provide an Async API URL



WebSocket API

Create WebSocket API

Create a Streaming API

Create an API by providing a Name, a Version, a Context and the Endpoint

Name*
EchoWebScoket

Context*
echo

Version*
1.0.0

API will be exposed in echo/1.0.0 context at the gateway

Protocol*
WebSocket

Endpoint
ws://echo.websocket.org:80

* Mandatory fields

Create

Create & Publish

Cancel

WebSocket API

Overview WebSocket API

The screenshot shows the API Manager interface with the following details:

API Overview: EchoWebSocket :1.0.0 (PUBLISHED)

Workflow Status: Last updated: a few seconds ago

Metadata:

- Description: -
- Provider: admin
- Context: /echo
- Version: 1.0.0
- Type: WebSocket
- Created Time: A few seconds ago
- Last Updated Time: A few seconds ago
- Business Owner: -
- Technical Owner: -

Configuration:

- Transports: OAuth2
- API Security: None
- Access Control: -
- Workflow Status: -
- Visibility on Developer Portal: Public
- Business Plans: AsyncUnlimited
- Tags: -

Topics:

- Sub: ^*
- PUB: ^*
- From User: ^*

Endpoints:

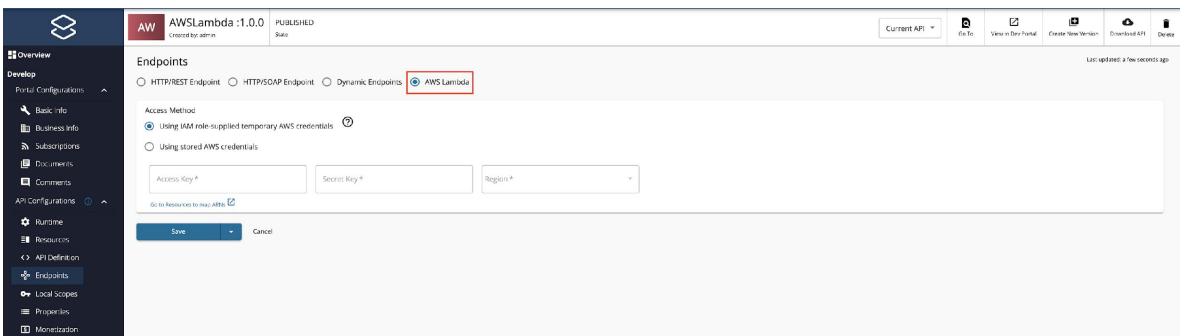
- Production Endpoint: ws://echo.websocket.org:80
- Sandbox Endpoint: ws://echo.websocket.org:80
- Endpoint Security: -

WebSocket API

Configuring endpoint

The screenshot shows the configuration page for the EchoWebSocket API version 1.0.0. The left sidebar includes sections for Overview, Develop, API Configurations (with endpoints selected), and Deploy. The main content area displays two endpoints: Production Endpoint (ws://echo.websocket.org:83) and Sandbox Endpoint (ws://echo.websocket.org:83). Both endpoints have checkboxes checked. The top right features standard API management controls like Current API, View in Dev Portal, Create New Version, Delete API, and a lock icon.

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.

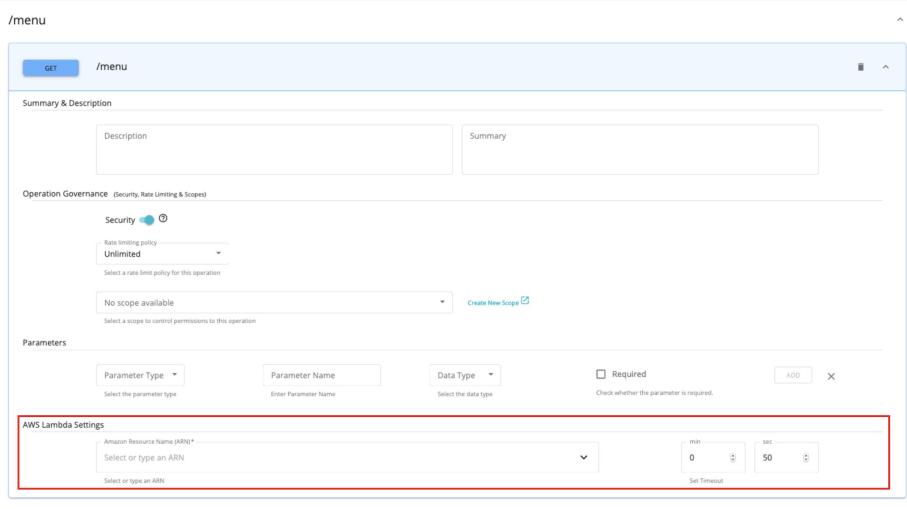
In AWS Lambda type, the credentials of the AWS API should be provided in this page.

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



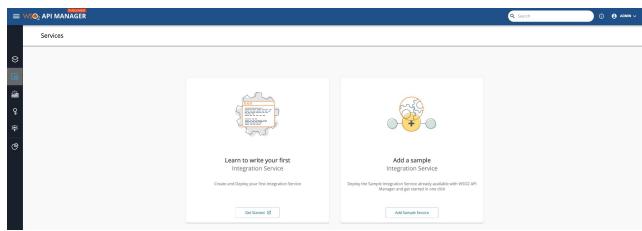
The screenshot shows the AWS Lambda configuration interface for selecting resource-level ARNs. It includes sections for Summary & Description, Operation Governance (Security, Rate Limiting & Scopes), Parameters, and AWS Lambda Settings. The AWS Lambda Settings section is highlighted with a red box, specifically around the 'Amazon Resource Name (ARN)' dropdown and the timeout settings.

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.

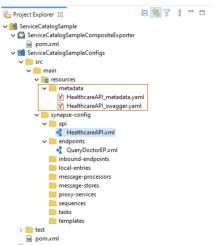
Create API Using a Service

- 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: <ul style="list-style-type: none">• Add the complete URL without parameters. For example: <code>http://localhost:8290/healthcare</code>.• Parameterize using the host and port combination. For example: <code>http://{{MI_HOST}}:{{MI_PORT}}/healthcare</code>.• Parameterize using a preconfigured URL. For example: <code>http://{{MI_URL}}/healthcare</code>.

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

3. Configure Micro Integrator Server
4. Start the servers

[Publish Integrations to API Manager](#)

Create an API Using a Service

1. Discover the service

The screenshot shows the WSO2 API Manager interface. On the left, a sidebar menu includes 'APIs', 'Services' (which is selected and highlighted in grey), 'API Products', 'Scopes', and 'Analytics'. The main area is titled 'Service Catalog' with 'Total: 1 Service'. A single service card is displayed, labeled 'Pi' with a blue background. Below it, the text 'Pizzashack-Endpoint...' is visible, followed by 'Version v1', 'Type OAS3', and two small circular icons. A red box highlights the 'Pi' service card.

2. Create API from the Service

The screenshot shows the 'Create API' dialog box overlaid on the Service Catalog. The dialog has a title 'Create API' and a sub-section 'Create API from service Pizzashack-Endpoint'. It contains a 'Name' field with 'Pizzashack-Endpoint' typed into it. Below the name is a 'Context' field containing '/api/v1'. A note below the context says 'API will be exposed in /templay/pizzashack/api/v1 context at API gateway'. At the bottom right of the dialog is a red-bordered 'Create API' button. The background shows the 'Pizzashack-Endpoint' service card with its details and a 'Create API' button on the right side of the card.

[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



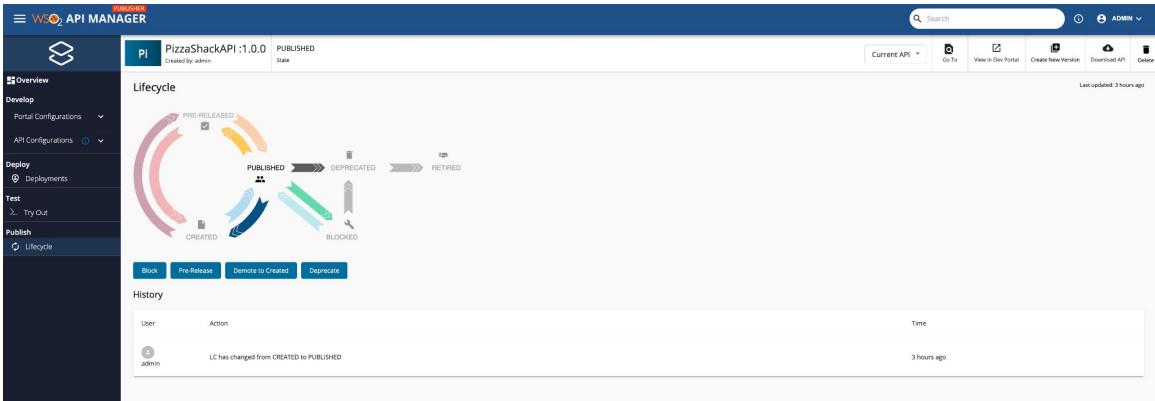
API Lifecycle

- 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.

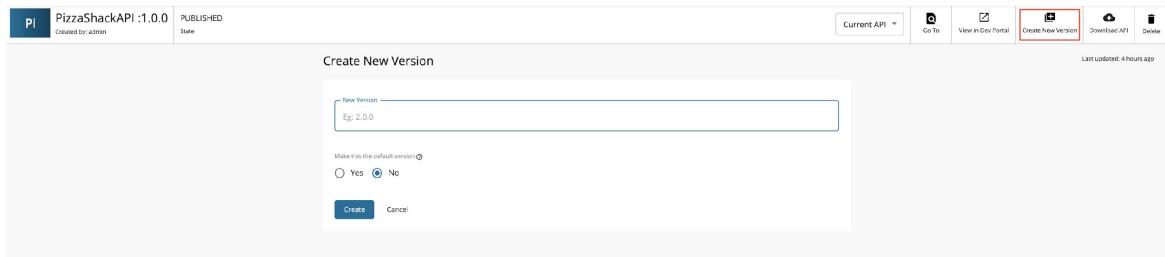
The screenshot shows the WSO2 API Manager Publisher interface. On the left, the navigation sidebar includes sections for Overview, Develop, API Configurations, Deploy (selected), Test, and Publish. Under Deploy, there are sub-options for Deployments and Lifecycle. The main content area displays the details for the 'PizzaShackAPI :1.0.0' API, which is in the 'PUBLISHED' state. A 'Deployments' section allows creating revisions and deploying them to gateway environments. A 'Revisions' timeline shows a single revision labeled 'Revision 1' (the latest) with options to 'Restore' or 'Delete'. Below this is an 'API Gateways' table:

Name	Gateway Access URL	Deployed Revision	Gateway URL Visibility
Default	http://localhost:8280 https://localhost:8243	Revision 1 Undeploy	Visible



API Versioning

Creating a new version of an existing API.



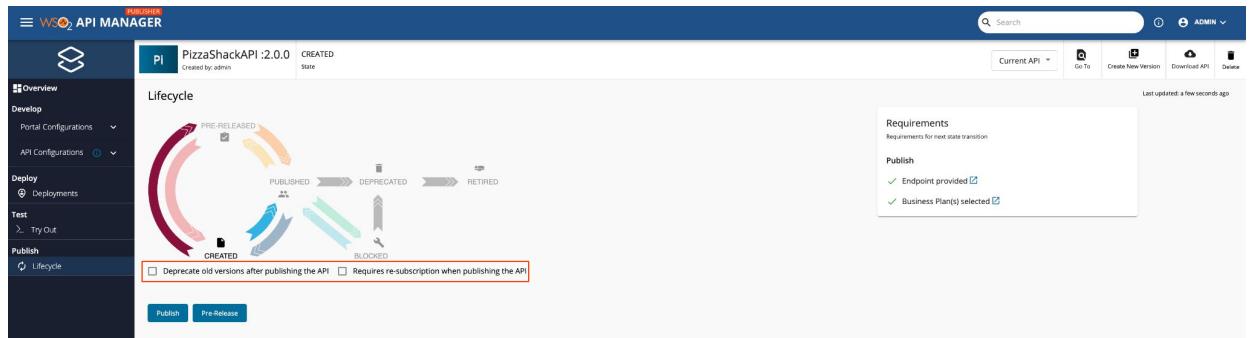
Select the API, and click “Create New Version” in the top panel.

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

The screenshot shows the 'Operations Configuration' section for managing API policies. It includes fields for 'Rate limiting level' (set to 'API Level'), 'HTTP Verb' (dropdown), 'URI Pattern' (text input), and a 'Policy' dropdown menu. The policy menu is open, displaying four options: '10KPerMin', '20KPerMin', '50KPerMin', and 'Unlimited'. A red box highlights this policy list. A blue '+' button is located to the right of the policy dropdown.

Throttling policies can be added per API or per resource.

More on Throttling policies in the Throttling section

Managing API Policies

Operation Level Policies

Resources

Last updated 14 minutes ago

Operations Configuration

Date limiting level API Level Operation Level

You may change the rate limiting policies per operation
Assign an operation below or click to set a rate limit policy for an operation.

HTTP Verb + X

/order

Root /order

Summary & Description

Description: Create a new Order

Scope:

Operation Governance (Security, Rate Limiting & Scopes)

Security

Rate limiting policy Unlimited Select a rate limit policy for this operation

Create New Scope

Select a scope to control permissions to this operation

The screenshot shows the 'Operations Configuration' section with 'Operation Level' selected. Below it is a table for defining rate limiting policies by HTTP verb. Under 'Operation Governance', the 'Security' tab is active, showing a dropdown for 'Rate limiting policy' with 'Unlimited' selected. A note says 'Select a rate limit policy for this operation'. The 'Operation scope' dropdown is also visible.

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 screenshot shows the WSO2 API Manager Publisher Portal. On the left, there's a sidebar with options like Overview, Develop, Deploy, Test (which is selected and highlighted with a red box), and Publish. Under Test, there's a sub-option 'Try Out'. The main area displays the 'PizzaShackAPI:1.0.0' API, which was created by 'admin'. It shows the status as 'PUBLISHED' and 'Last updated: 2 minutes ago'. Below this, there's a 'Security' section with a large, illegible security key. A 'General Key' button is also present. The 'API Gateways' section shows a single entry for 'Default'. Under 'Servers', it lists 'https://localhost:8243/pizzashack/1.0.0'. The 'default' configuration pane shows five API endpoints: POST /order, GET /menu, GET /order/{orderId}, PUT /order/{orderId}, and DELETE /order/{orderId}. Each endpoint is color-coded (green for POST, blue for GET, orange for PUT, red for DELETE).

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

The screenshot shows a Swagger UI interface for testing APIs. A modal window is open for a GET request to the endpoint `/menu`. The modal has tabs for "Parameters" (which shows "No parameters"), "Responses", and "Code". Under "Responses", the status code 200 is selected, and the "Response body" section displays the following JSON data:

```
[{"name": "BBQ Chicken Bacon", "description": "Grilled white chicken, hickory-smoked bacon and fresh sliced onions in barbecue sauce", "price": "23.99", "icon": "/image/6.png"}, {"name": "Chicken Parmesan", "description": "Grilled chicken, fresh tomatoes, feta and mozzarella cheese", "price": "19.99", "icon": "/image/1.png"}, {"name": "Chili Chicken Cordon Bleu", "description": "Spicy chili Alfredo sauce topped with grilled chicken, ham, onions and mozzarella", "price": "19.99", "icon": "/image/10.png"}, {"name": "Double Bacon G'Cheese", "description": "Hickory-smoked bacon, Julienne cut Canadian bacon, Parmesan, mozzarella, Romano, Asiago and Fontina cheese", "price": "18.99", "icon": "/image/9.png"}]
```

Below the JSON, there are sections for "Response headers" (with `content-type: application/json`) and "Code Details". At the bottom right of the modal, there are "Download" and "Copy" buttons.

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

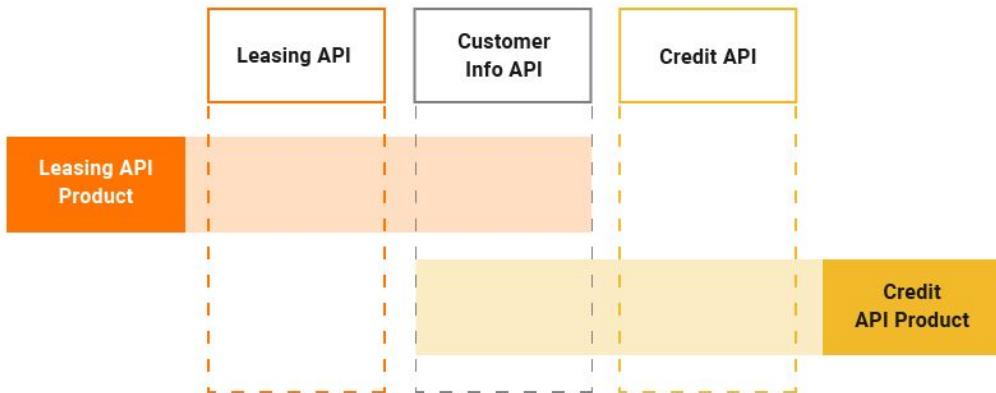
test-key token - If an API developer initiates a test, the swagger console will be automatically populated with the generated test-key. If you try the test API call through the terminal or command line, make sure you copy the generated test-key.

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



Create an API Product
Create an API Product by providing a Name, a Context, Resources, and Business Plans (optional).

This is a screenshot of the 'Create an API Product' wizard, Step 1: Define API Product. It shows two input fields: 'Name*' containing 'CustomerLeasing' and 'Context*' containing 'customerleasing'. A note below says 'API Product will be exposed in customerleasing(v1) context at the gateway'. There are 'Cancel' and 'Next' buttons at the bottom. A small note indicates '* Mandatory fields'.

- 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

Create an API Product
Create an API Product by providing a Name, a Context, Resources, and Business Plans (optional).

Select an API Select API Resources

Define API Product Add Resources

API
Filter APIs
Filter by name

customer-info
1.0.0 - /customerinfo

leasing
1.0.0 - /leasing

leasing

Add Selected ►| Add All ►| Back | Publish

CustomerLeasing

customer-info - 1.0.0
GET /customers
GET /customers/{customerId}

leasing - 1.0.0
POST /assets
GET /assets/{assetId}

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

The screenshot shows the 'customer-leasing' API product in the 'CREATED' state. The left sidebar includes 'Overview', 'Develop', 'Deploy', 'Test', and 'Publish' sections. The main area displays a lifecycle diagram with four stages: 'Develop' (green), 'Deploy' (blue), 'Test' (grey), and 'Published' (grey). Below the diagram, the 'Context:' and 'Configuration' sections are listed.

Context:		Configuration:	
Description	-	Transports	HTTP, HTTPS
Provider	Publisher	API Security	OAuth2
Context:	/customer-leasing	Access Control	None
Created Time	A few seconds ago	Workflow Status	-
Last Updated Time	A few seconds ago	Visibility on Developer Portal	Public
Business Owner	-	Business Plans:	Unlimited
Technical Owner	-		

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

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

- Header:** WSO2 API Manager, PizzaShackAPI:1.0.0 (Published), Search bar, CMS (highlighted in red), READ ONLY (highlighted in red).
- Left Sidebar:** Overview, Develop, Deploy, Test. Under Develop, Portal Configurations is expanded, showing Basic Info, Business Info, Subscriptions, Documents, and Comments.
- Current API:** Current API dropdown, Go To, View in Dev Portal, Download API.
- Right Content Area:** Design Configurations for PizzaShackAPI:1.0.0. The dialog is titled "Design Configurations".
 - Publisher Access Control:** Set to "All".
 - Developer Portal Visibility:** Set to "Public".
 - Tags:** A single tag "pizza" is listed.
 - API Categories:** No API Categories defined.
 - GitHub URL:** Placeholder for GitHub URL.
 - Slack URL:** Placeholder for Slack URL.
 - Mark the API as third party:** Radio buttons for "This" (unchecked) and "No" (checked).
 - Active Deployments:** A note: "There are active deployments in the API. Please complete the revision before changing the API to a third party API." (with a blue background).
 - Make this the default version:** Radio buttons for "Yes" (unchecked) and "No" (checked).

Let's try it out!

Creating and publishing an API Product with API Manager Publisher

