



WSO2 API Manager 3.2.0

Fundamentals

Getting Started with the API Publisher

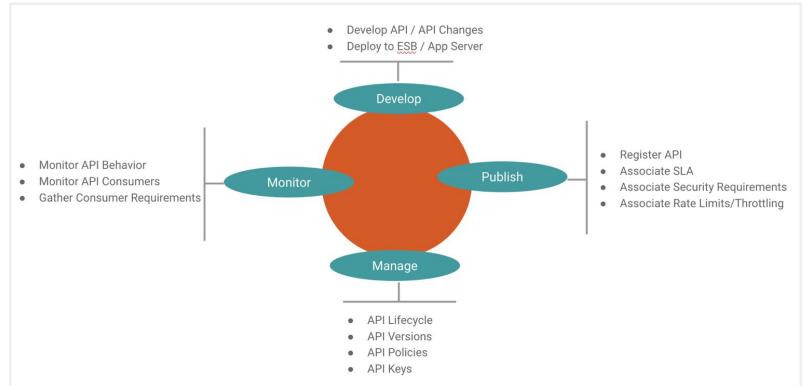


WSO2 Training



What is the API Publisher Portal?

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



Creating an API



This section covers,

1. How to create a REST API from scratch
2. Complete UI walkthrough
3. Create other API Types

Let's try it out!

Creating and publishing an API through API Manager Publisher



Ways of Creating APIs

- Design a new REST API
- REST API from OpenAPI definition
- REST API from SOAP backend
- SOAP API as a REST service
- GraphQL API from SDL
- Design new Websocket API

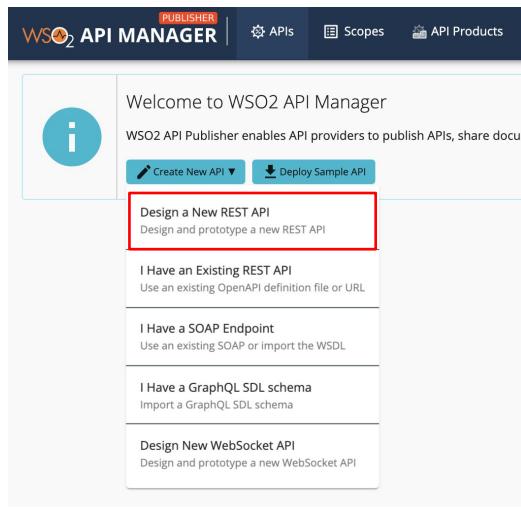
[Creating APIs](#)

The screenshot shows the WSO2 API Manager Publisher interface. At the top, there's a navigation bar with tabs for 'PUBLISHER' (highlighted), 'APIs', 'Scopes', and 'API Products'. Below the navigation bar, there's a large blue 'i' icon and a welcome message: 'Welcome to WSO2 API Manager'. The message states: 'WSO2 API Publisher enables API providers to publish APIs, share documentation, and manage API products'. Below this, there are four main creation options:

- Create New API ▾**: 'Design a New REST API' (Use an existing OpenAPI definition file or URL)
- I Have an Existing REST API**: 'Use an existing OpenAPI definition file or URL'
- I Have a SOAP Endpoint**: 'Use an existing SOAP or Import the WSDL'
- I Have a GraphQL SDL schema**: 'Import a GraphQL SDL schema'
- Design New WebSocket API**: 'Design and prototype a new WebSocket API'

Design a New REST API

Select the **Design a New REST API** 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

Create an API

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

The screenshot shows a form for creating an API. The fields filled in are:

- Name*: PizzaShack
- Context*: /pizzashack
- Version*: 1.0.0
- Endpoint: https://localhost:9443/am/sample/pizzashack/v1/api/ (with a checked checkbox)
- Business plan(s): Unlimited

Below the form, it says "Select one or more throttling policies for the API". A note at the bottom right indicates "* Mandatory fields". At the bottom of the form are three buttons: CREATE, CREATE & PUBLISH (which is highlighted with a red box), and CANCEL.

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. At the top, there's a navigation bar with links for APIs, Scopes, API Products, and a search bar. Below the navigation is a header for the "PizzaShackAPI:1.0.0" API, showing it was created by "admin" and is in the "CREATED" state. On the right side of the header are buttons for Go To, Create New Version, Download API, and Delete. A message at the top right says "Last updated: a few seconds ago".

The main content area is divided into several sections:

- Overview:** Shows a progress bar with three steps: "Created" (checkmark), "Endpoint" (checkmark), and "Business plans" (checkmark). A button labeled "Published" is shown in a blue box.
- Metadata:** Describes the API as "This is a simple API for Pizza Shack online pizza delivery store." It includes fields for Provider (admin), Context (/pizzashack), Version (1.0.0), Type (HTTP), Created Time (A few seconds ago), Last Updated Time (A few seconds ago), Business Owner (Jane Roe), and Technical Owner (John Doe).
- Configuration:** Lists transports (HTTP, HTTPS), API Security (OAuth2), Access Control (None), Workflow Status (None), and Visibility on Developer Portal (Public). It also shows Business Plans (Unlimited) and Tags (pizza).
- Endpoints:** Provides the Production Endpoint (https://localhost:9443/api/sample/pizzashack/v1/api) and Sandbox Endpoint (https://localhost:9443/api/sample/pizzashack/v1/api). It also lists Endpoint Security.
- Resources:** Lists three resources: "order" (with a green "POST" icon), "menu" (with a blue "GET" icon), and "order/{orderId}" (with a blue "GET" icon, a yellow "PUT" icon, and a red "DELETE" icon). There is also a "Show More" link.

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.

The screenshot shows the WSO2 API Manager interface. On the left, a sidebar lists various API management components like Overview, Design Configurations (which is highlighted with a red box), Runtime Configurations, Resources, Endpoints, Subscriptions, Lifecycle, API Definition, Environments, Local Scopes, Business Info, Properties, Documents, Test Console, and Monetization. The main content area is titled 'Design Configurations' and contains fields for 'Description' (a simple API for Pizza Shack online pizza delivery store), 'Publisher Access Control' (set to 'All'), 'Developer Portal Visibility' (set to 'Public'), 'Tags' (containing 'pizza'), and 'API Categories' (empty). At the bottom, there's a section to 'Make this the default version' with radio buttons for 'Yes' (selected) and 'No'. A 'Save' button is at the bottom right.

Design Configuration includes,

- **Publisher Access control** in the Publisher portal
- **Developer Portal Visibility** in the Developer Portal
- **Tags**
- **API Categories**
- **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 Runtime Configurations

The screenshot shows the WSO2 API Manager interface. In the top navigation bar, 'APIs' is selected. On the left sidebar, 'Runtime Configurations' is highlighted with a red box. The main content area displays the configuration for the 'PizzaShackAPI - 1.0.0' API, which was created by 'admin' and is in 'CREATED' state. The configuration is divided into three sections: Request, Response, and Fault. Under Request, there are dropdowns for Transport Level Security, Application Level Security, CORS Configuration, Schema Validation, and Message Mediation (set to 'none'). Under Response, there are dropdowns for Message Mediation and Response Caching (set to 'none'). Under Fault, there is a dropdown for Message Mediation (set to 'none'). On the right side, there is a 'Backend' section with 'Backend Throughput' settings (Maximum Throughput: Unlimited) and an 'Endpoints' section listing 'Production' and 'Sandbox' endpoints. A 'Save' button is at the bottom left, and a 'Cancel' button is at the bottom right.

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
- **Message Mediation** Engage request transformation policies.

Response: How the Response flow should be handled.

- **Message Mediation** Engage Response transformation policies
- **Response Caching** Set Response caching configuration

Fault: How to handle any error situation

- **Message Mediation** Engage Fault Flow Mediation policy.

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.

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

The screenshot shows the WSO2 API Manager interface. The left sidebar is dark with white icons and text. The 'Endpoints' icon is highlighted with a red box. The main content area shows a list of endpoints for the 'PizzaShackAPI'. There are two entries: 'Production Endpoint' and 'Sandbox Endpoint'. Below the list, there are sections for 'General Endpoint Configurations' and 'Load balance and Failover Configurations'. At the bottom of the main content area, there are 'Save' and 'Cancel' buttons. A modal window is overlaid on the main content, also titled 'Endpoints'. It contains five options with radio buttons: 'HTTP/ REST Endpoint', 'HTTP/ SOAP Endpoint', 'Prototype Endpoint', 'Dynamic Endpoint', and 'AWS Lambda Endpoint'. The 'HTTP/ REST Endpoint' option is selected and highlighted with a red box.

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)
- **Prototype Endpoint**
 - **Prototype Endpoint** Mock endpoint
 - **Prototype Implementation** Use inbuilt javascript engine to mock the api response
- **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, a sidebar menu lists various options: Overview, Design Configurations, Runtime Configurations, Resources, Endpoints, Subscriptions (which is highlighted with a red box), Lifecycle, API Definition, Environments, Local Scopes, Business Info, Properties, Documents, Test Console, and Monetization. The main content area displays the 'Business Plans' section for the 'PizzaShackAPI :1.0.0' API. It shows a list of subscription plans: Bronze (Allows 1000 requests per minute), Gold (Allows 5000 requests per minute), Silver (Allows 2000 requests per minute), and Unlimited (Allows unlimited requests). The 'Unlimited' plan has a checked checkbox. Below this is a 'Manage Subscriptions' section which states 'No subscriptions data available'. At the top right of the main content area, there are buttons for Go To, Create New Version, Download API, and Delete. The top navigation bar includes links for APIs, Scopes, API Products, a search bar, and settings/admin options. A footer at the bottom left indicates 'WSO2 API-M v3.2.0 | © 2020 WSO2 Inc'.

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 with the 'API Definition' page selected. The main content area displays the API definition in YAML format. Several sections are highlighted with red boxes:

- The top navigation bar has a red box around the 'API Definition' tab.
- The main content area has a red box around the entire code block.
- A modal window titled 'UPDATE CONTENT' also has a red box around its content area.
- Specific sections within the code block are highlighted with red boxes:
 - The 'default' section under 'paths'.
 - The 'POST /order' section under 'operations'.
 - The 'POST /order' section within the 'UPDATE CONTENT' modal.

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

Gateway Environments

The screenshot shows the WSO2 API Manager interface. On the left, a sidebar lists various management options: Overview, Design Configurations, Runtime Configurations, Resources, Endpoints, Subscriptions, Lifecycle, API Definition, **Environments** (which is highlighted with a red box), Local Scopes, Business Info, Properties, Documents, Test Console, and Monetization. The main content area displays the details for the 'PizzaShackAPI :1.0.0' API, which is in a 'CREATED' state and was created by 'admin'. Below this, the 'API Gateways' section lists a single entry: 'Production and Sandbox' (Type: hybrid) with Server URLs 'https://localhost:9443/services/' and 'http://localhost:8280', and HTTPS URL 'https://localhost:8243'. The 'Gateway Labels' section indicates that no labels are available, with a note: 'Gateway labels are not available. You can request the administrator to add labels.' At the bottom of the page, there are 'Save' and 'Cancel' buttons.

Select the required Gateway Environment this API should be deployed to. Since there are no other environments, select the default Production and Sandbox environment.

When publishing the API, it will be published to all the selected environments.

Local Scopes

The screenshot shows the WSO2 API Manager interface. In the top navigation bar, 'Scopes' is selected. Below it, the 'PizzaShackAPI:1.0.0' API is listed with a 'CREATED' status. On the left sidebar, the 'Local Scopes' option is highlighted with a red box. The main content area shows a form titled 'Create New Scope' with the following fields:

- Name: orderPizza
- Description: Order Pizza
- Notes: admin

At the bottom of the form 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 interface. At the top, there is a navigation bar with tabs for PUBLISHER, APIs, Scopes (which is highlighted with a red box), and API Products. A search bar is also present. Below the navigation bar, a large callout box on the left contains an 'i' icon and the text "Welcome to WSO2 API Manager" followed by "Scopes enable fine-grained access control to API resources based on user roles." A red box highlights the "Create a new scope" button. The main content area has a breadcrumb navigation "Scopes > Create New Scope". The form fields are as follows:

- Name: CreateScope (highlighted with a red box)
- Display Name: CreateSharedScope
- Description: Restricting create operations to users with admin role
- Roles:
 - admin (highlighted with a red box)

At the bottom of the form are "Save" and "Cancel" buttons, with "Save" highlighted with a red box.

In WSO2 API-M, an OAuth scope can be created before the API is created and 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 (name and email) and technical owner details (name and email). It includes "SAVE" and "CANCEL" buttons.
- API Properties:** A form for defining custom properties (Property Name and Value) such as "co-owner" and "John". It includes "ADD PROPERTY" and "CANCEL" buttons.
- Documents:** A form for adding new documents, requiring a name, summary, type (e.g., How To, Sample & SDK), source (e.g., Inline, Markdown, URL, File), and a note to save the document. It includes "ADD DOCUMENT" and "CANCEL" buttons.

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

What is Swagger/OpenAPI ?

Create an API using an OpenAPI definition.

Create an API using an existing OpenAPI definition (swagger) file or URL.

Provide OpenAPI

Create API

* Input Type
 OpenAPI URL
 OpenAPI File

OpenAPI URL
http://petstore.swagger.io/v2/swagger.json ✓

Click away to validate the URL.

CANCEL NEXT

- 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/			
[GET]	/pet/{petId}	Find pet by ID	-
[POST]	/pet/{petId}	Updates a pet in the store with form data	-
[DELETE]	/pet/{petId}	Deletes a pet	-
/pet/findByStatus			
[POST]	/pet/findByStatus	uploads an image	-
/pet			
[POST]	/pet	Add a new pet to the store	-
[PUT]	/pet	Update an existing pet	-
/pet/findByTags			
[GET]	/pet/findByTags	Finds Pets by tags	-
/store/inventory			
[GET]	/store/inventory	Returns pet inventories by status	-
/store/order/{orderId}			
[GET]	/store/order/{orderId}	Find purchase order by ID	-
[DELETE]	/store/order/{orderId}	Delete purchase order by ID	-
/store/order			
[POST]	/store/order	Place an order for a pet	-
/user/username			
[GET]	/user/username	Get user by user name	-
[PUT]	/user/username	Updated user	-
[DELETE]	/user/username	Delete user	-
/user/login			
[GET]	/user/login	Logs user into the system	-

REST API from OpenAPI Definition

Resources can be edited inline

```
1  -- 
2  swagger: "2.0"
3  info:
4    description: "This is a sample server Petstore server. You can find out more about
5      Swagger at [http://swagger.io](http://swagger.io) or on [irc.freenode.net, #swagger]
6      ([irc://irc.freenode.net/#!/#swagger]). For this sample, you can use the api key special-key
7      to test the authorization filters."
8
9    version: "1.0.3"
10   title: "Swagger Petstore"
11   termsOfService: "http://swagger.io/terms/"
12   contact:
13     email: "optteam@swagger.io"
14   license:
15     name: "Apache 2.0"
16     url: "http://www.apache.org/licenses/LICENSE-2.0.html"
17   host: "petstore.swagger.io"
18   basePath: "/v2"
19   tags:
20
21     name: "pet"
22     description: "Everything about your Pets"
23     externalDocs:
24       description: "Find out more"
25       url: "http://swagger.io"
26
27     name: "store"
28     description: "Access to Petstore orders"
29
30     name: "user"
31     description: "Operations about user"
32
33   schemes:
34     - "https"
35     - "http"
36
37   security:
38     default: □
39
40   paths:
41     /pet/{petId}:
```

Swagger Petstore 1.0.3

This is a sample server Petstore server. You can find out more about Swagger at <http://swagger.io> or on [irc.freenode.net, #swagger](#). For this sample, you can use the api key `special-key` to test the authorization filters.

Terms of service
Contact the developer
Apache 2.0
Find out more about Swagger

Schemes HTTPS HTTP Auth

pet Everything about your Pets Find out more: <http://swagger.io>

Method	Path	Description	Auth
GET	/pet/{petId}	Find pet by ID	🔒
POST	/pet/{petId}	Updates a pet in the store with form data	🔒
DELETE	/pet/{petId}	Deletes a pet	🔒
POST	/pet/{petId}/uploadImage	uploads an image	🔒
POST	/pet	Add a new pet to the store	🔒

REST API from OpenAPI Definition

Configure Production and Sandbox endpoints

Endpoints

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

Production Endpoint

Production Endpoint *

http://petstore.swagger.io/v2/

Sandbox Endpoint

General Endpoint Configurations

Endpoint Security : NONE | Certificates: 0

Load balance and Failover Configurations

Not Configured

SAVE

CANCEL



REST API from OpenAPI Definition

Configure Business Plans in Subscription Tab

The screenshot shows the WSO2 API Manager interface. The top navigation bar includes links for PUBLISHER, APIs, Scopes, API Products, a search bar, and a help icon. The main area displays the 'SwaggerPetstore :1.0.5' API, which was 'Created by: admin' and is currently in 'CREATED' state. On the left, a sidebar menu lists various management options: Overview, Design Configurations, Runtime Configurations, Resources, Endpoints, Subscriptions (which is selected), Lifecycle, API Definition, Environments, Local Scopes, and Business Info. The 'Business Plans' section allows attaching business plans to the API. It lists four options: Bronze (Allows 1000 requests per minute), Gold (Allows 5000 requests per minute, checked), Silver (Allows 2000 requests per minute, checked), and Unlimited (Allows unlimited requests). Below this are 'Save' and 'Cancel' buttons. A second section titled 'Manage Subscriptions' is also visible.

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.



REST API from SOAP Backend

The screenshot shows the WSO2 API Manager interface. At the top, there's a navigation bar with tabs for PUBLISHER, APIs, Scopes, and API Products. Below the navigation bar, there's a sidebar titled "APIs" which says "Displaying 1 API(s)" and shows a thumbnail for "PizzaShackAPI". To the right of the sidebar, there's a "CREATE API ▾" button. The main content area has several options:

- "Design a New REST API": "Design and prototype a new REST API"
- "I Have an Existing REST API": "Use an existing OpenAPI definition file or URL"
- "I Have a SOAP Endpoint": "Use an existing SOAP or import the WSDL" (This option is highlighted with a red box)
- "I Have a GraphQL SDL schema": "Import a GraphQL SDL schema"
- "Design New WebSocket API": "Design and prototype a new WebSocket API"

At the bottom left of the main content area, there's a "PUBLISHED" button. On the far left and right edges of the interface, there are small circular icons.

REST API from SOAP Backend

Two options to create APIs for SOAP backends

- Pass Through
- Generate REST APIs

Expose a SOAP Service as a REST API
Expose an existing SOAP service as a REST API by importing the WSDL of the SOAP service.

Provide WSDL Create API

① ————— ②

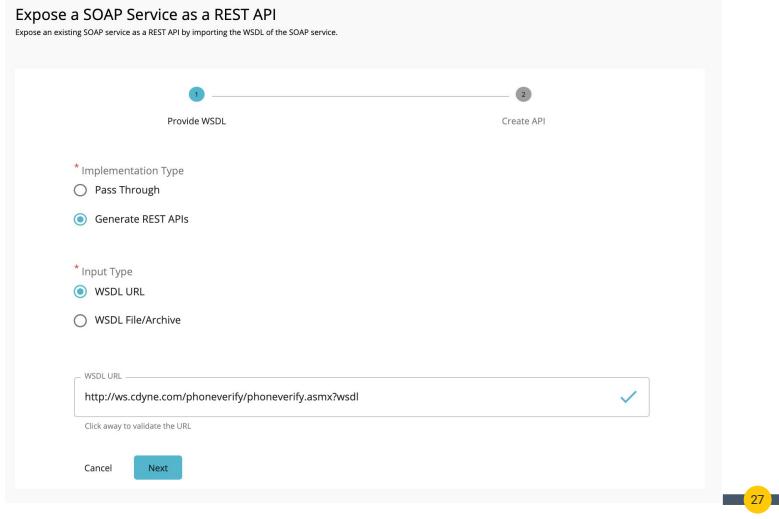
* Implementation Type
 Pass Through
 Generate REST APIs

* Input Type
 WSDL URL
 WSDL File/Archive

WSDL URL
http://ws.cdyne.com/phoneverify/phoneverify.asmx?wsdl ✓

Click away to validate the URL

Cancel Next



REST API from SOAP Backend

API Creation Menu

Expose a SOAP Service as a REST API

Expose an existing SOAP service as a REST API by importing the WSDL of the SOAP service.

Provide WSDL

Create API

Name^{*}
PhoneVerification

Context^{*}
/phoneverify

Version^{*}
1.0.0

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

Endpoint
<http://ws.cdyne.com/phoneverify/phoneverify.asmx?wsdl>

Business plan(s)
Unlimited

* Mandatory fields

BACK

CREATE



28

REST API from SOAP Backend

API Overview Page

The screenshot shows the WSO2 API Manager interface for the "PhoneVerification :1.0" API. The top navigation bar includes links for APIs, Scopes, API Products, a search bar, and settings like SETTINGS and ADMIN.

The main area displays the API's status as "CREATED" and its provider as "admin". A timeline indicates the API was "Created" and has since had an "Endpoint" added. A "Publish" button is available.

Metadata:

- Description: -
- Provider: admin
- Context: /phoneverify
- Version: 1.0
- Type: SOAPTOREST
- Created Time: A few seconds ago
- Last Updated Time: A few seconds ago
- Business Owner: -
- Technical Owner: -

Configuration:

- Transports: HTTP, HTTPS
- API Security: OAuth2
- Access Control: None
- Workflow Status: -
- Visibility on Developer Portal: Public
- Business Plans: Unlimited
- Tags: -

Endpoints:

- Production Endpoint: <http://ws.cdyne.com/phoneverify/phoneverify.asmx?wsdl>
- Sandbox Endpoint: <http://ws.cdyne.com/phoneverify/phoneverify.asmx?wsdl>
- Endpoint Security: -

Resources:

- /checkPhoneNumbers (POST)
- /checkPhoneNumber (POST)

A "Show More" link is also present.

29

REST API from SOAP Backend

In/Out Sequences of resources

The screenshot shows a configuration interface for a REST API endpoint named "/checkPhoneNumber". The "In" sequence is highlighted with a red box. The sequence details are as follows:

- Description:** Description: checkPhoneNumbers
- Rate Limiting:** Rate limiting is governed by API Level.
- Operation scope:** Select a scope to control permissions to this operation.
- Meditation:** The "In" sequence is shown with the following code:

```
1 <ns0:checkPhoneNumbers xmlns:ns0="http://www.w3.org/2005/08/soap-envelope">
2   <ns0:to>http://cognitiveservices.dynamilis.com/CheckPhoneNumbers</ns0:to>
3   <ns0:from>http://cognitiveservices.dynamilis.com/CheckPhoneNumbers</ns0:from>
4   <ns0:action>http://cognitiveservices.dynamilis.com/CheckPhoneNumbers</ns0:action>
5   <ns0:headers></ns0:headers>
6   <ns0:body></ns0:body>
7   <ns0:use>factory</ns0:use>
8   <ns0:transform>urn:xsd:soap:envelope</ns0:transform>
9   <ns0:inSequence>
10    <ns0:header></ns0:header>
11    <ns0:operation>
12      <ns0:method>get</ns0:method>
13      <ns0:uri>http://cognitiveservices.dynamilis.com/CheckPhoneNumbers</ns0:uri>
14      <ns0:withPhoneNumbers>$2</ns0:withPhoneNumbers>
15      <ns0:withPhoneNumbers>1234567890</ns0:withPhoneNumbers>
16    </ns0:operation>
17    <ns0:logBody>
18      <ns0:logBody></ns0:logBody>
19    </ns0:logBody>
20    <ns0:args>
21      <ns0:arg>
22        <ns0:expression>get-property("req:var.CheckPhoneNumbers.LicenseKey")</ns0:expression>
23        <ns0:evaluator>xsl</ns0:evaluator>
24        <ns0:property>req:var.CheckPhoneNumbers.LicenseKey</ns0:property>
25      </ns0:arg>
26    </ns0:args>
27    <ns0:payload></ns0:payload>
28    <ns0:property>messageProperties</ns0:property>
29      <ns0:property>messageType</ns0:property>
30        <ns0:value>application/xop+xml</ns0:value>
31        <ns0:type>STRING</ns0:type>
32      </ns0:property>
33    </ns0:property>
34  </ns0:inSequence>
35</ns0:checkPhoneNumbers>
```



SOAP API as a REST Service

Expose a SOAP Service as a REST API

Expose an existing SOAP service as a REST API by importing the WSDL of the SOAP service.

Provide WSDL Create API

* Implementation Type
 Pass Through
 Generate REST APIs
*Generate REST APIs option is only available for WSDL URL input type

* Input Type
 WSDL URL
 WSDL File/Archive

WSDL URL
http://ws.cdyne.com/phoneverify/phoneverify.asmx?wsdl ✓

Click away to validate the URL

CANCEL NEXT



SOAP API as a REST Service

SOAP API Creation

Expose a SOAP Service as a REST API
Expose an existing SOAP service as a REST API by importing the WSDL of the SOAP service.

Provide WSDL

Create API

Name*

Context* API will be exposed in /phonverify/1.0.0 context at the gateway

Version*

Endpoint

Business plan(s)

Select one or more throttling policies for the API

* Mandatory fields

BACK

SOAP API as a REST Service

API Overview

The screenshot shows the WSO2 API Manager interface for the 'PhoneVerification:1.0.0' API. The top navigation bar includes 'APIs', 'API Products', 'Search', 'SETTINGS', and 'ADMIN'. The main content area displays the API's state as 'CREATED' and its version as '1.0.0'. A progress bar indicates the API is 'Created' and has '✓ Endpoint' and '✓ Business plans' checked, with a 'PUBLIC' status. The left sidebar lists various management sections: Overview, Design Configurations, Runtime Configurations, Resources, Endpoints, Subscriptions, Lifecycle, API Definition, Environments, Scopes, Business Info, Properties, Documents, and Monetization.

Metadata		Configuration	
Description	-	Transports	HTTP, HTTPS
Provider	admin	API Security	OAuth2
Context	/phonoverify	Access Control	None
Version	1.0.0	Workflow Status	-
Type	SOAP	Visibility on Developer Portal	Public
Created Time	2 hours ago	Business Plans	Unlimited
Last Updated Time	2 hours ago	Tags	-
Business Owner	-	Endpoints	
Technical Owner	-	Production Endpoint	http://ws.cdyne.com/phonoverify/phonoverify.asmx?wsdl
Resources	REST	Sandbox Endpoint	http://ws.cdyne.com/phonoverify/phonoverify.asmx?wedi
Show More			

GraphQL API from SDL

The screenshot shows the WSO2 API Manager interface. At the top, there's a navigation bar with tabs for 'PUBLISHER' (highlighted), 'APIs', 'Scopes', and 'API Products'. Below the navigation bar, there's a sidebar titled 'APIs' with the subtext 'Displaying 1 API(s)'. A large button labeled 'CREATE API ▾' is visible. To the right of the sidebar, there are several options for creating APIs:

- 'Design a New REST API': 'Design and prototype a new REST API'
- 'I Have an Existing REST API': 'Use an existing OpenAPI definition file or URL'
- 'I Have a SOAP Endpoint': 'Use an existing SOAP or import the WSDL'
- 'I Have a GraphQL SDL schema': 'Import a GraphQL SDL schema' (this option is highlighted with a red box)
- 'Design New WebSocket API': 'Design and prototype a new WebSocket API'

On the left side of the main content area, there's a card for a published API named 'PizzaShackAPI'. The card includes details like 'By : admin', '1.0.0', 'Version', '/pizzashack', 'Context', and a 'PUBLISHED' status indicator.

GraphQL, which has been developed by Facebook, is a data query language for APIs. When using GraphQL, users can explicitly specify as to what data they need from an API. GraphQL APIs are an alternative to REST-based APIs.

GraphQL API from SDL

Import SDL

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

Provide GraphQL

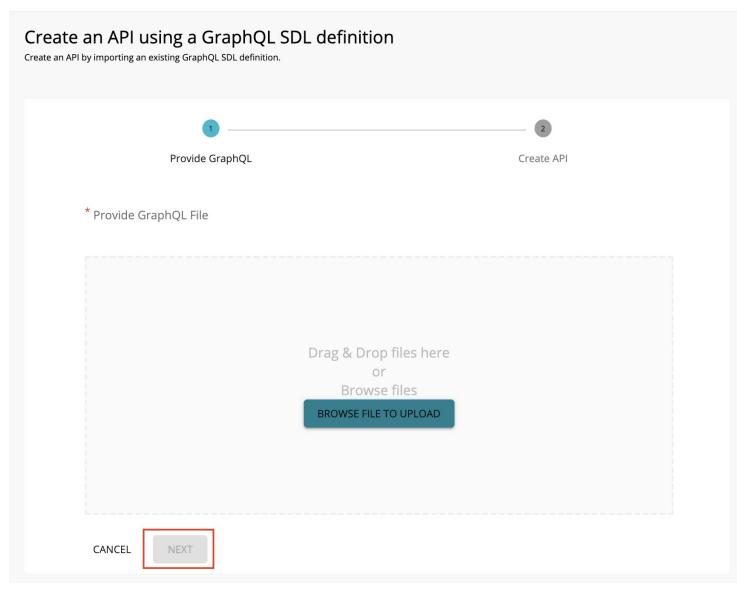
2 Create API

* Provide GraphQL File

Drag & Drop files here
or
Browse files

BROWSE FILE TO UPLOAD

CANCEL NEXT



You can use a Schema Definition Language (SDL) schema to design a GraphQL API in WSO2 API Manager (WSO2 API-M) similar to creating SOAP APIs using WSDLs and developing REST APIs using OpenAPI Specifications (a.k.a. Swagger Definitions).

GraphQL API from SDL

Import SDL

Create an API using a GraphQL SDL definition

Create an API by importing an existing GraphQL SDL definition.

1

Provide GraphQL

2

Create API



schema_graphql.graphql - 190.0 KiB



CANCEL

NEXT



36

GraphQL API from SDL

Create API

Create an API using GraphQL SDL definition

Use an existing GraphQL SDL definition file to create an API in WSO2 API Manager.

Provide GraphQL 2 Create API

Name*
StarWarsAPI

Context*
/swapi

Version*
1.0.0

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

Endpoint
<https://api.graph.cool/simple/v1/swapi>

Business plan(s)
Unlimited

Select one or more throttling policies for the API Product

* Mandatory fields

BACK CREATE

37

GraphQL API from SDL

API Overview

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

Header: PUBLISHED, APIs, Scopes, API Products, Search, SETTINGS, ADMIN.

API Overview: StarWarsAPI :1.0.0, CREATED, State: State.

Workflow Status: Created → Endpoint → Business plans → Publish.

Metadata:

Description	Value
Provider	admin
Context	/swapi
Version	1.0.0
Type	GRAPHQL
Created Time	A few seconds ago
Last Updated Time	A few seconds ago
Business Owner	-
Technical Owner	-

Configuration:

Setting	Value
Transports	HTTP, HTTPS
API Security	OAuth2
Access Control	None
Workflow Status	-
Visibility on Developer Portal	Public
Business Plans	Unlimited
Tags	-

Operations:

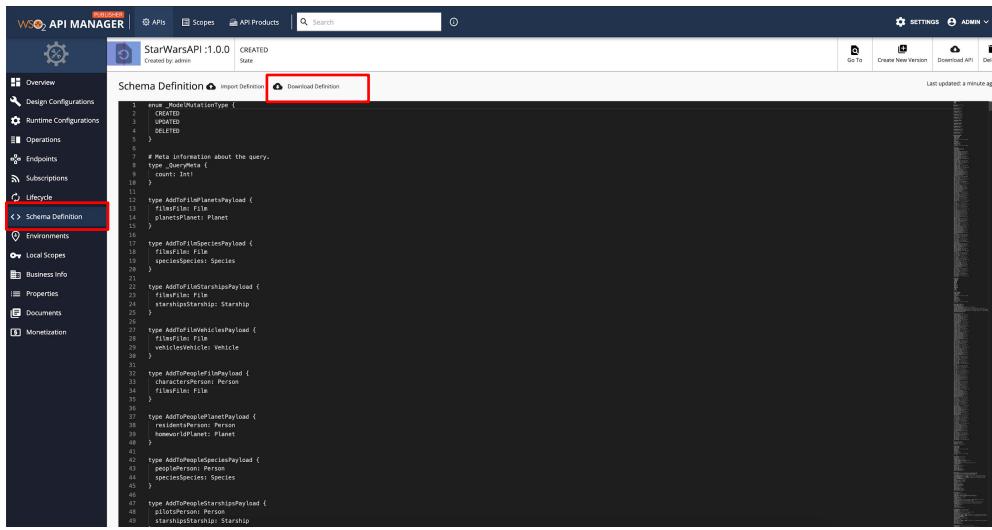
Operation	Type
allFilms	QUERY
allVehicles	QUERY
Film	QUERY
updateOrCreateFilm	MUTATION
deleteSpecies	MUTATION
_allVehiclesMeta	QUERY
removeFromPeopleVehicles	MUTATION

Endpoints:

Endpoint Type	URL
Production Endpoint	https://api.graph.cool/simple/v1/swapi
Sandbox Endpoint	https://api.graph.cool/simple/v1/swapi
Endpoint Security	-

GraphQL API from SDL

Edit SDL Schema



The screenshot shows the WSO2 API Manager interface with the "StarWarsAPI:1.0.0" API selected. The left sidebar includes options like Overview, Design Configurations, Runtime Configurations, Operations, Endpoints, Subscriptions, Lifecycle, Schema Definition (which is highlighted with a red box), Environments, Local Scopes, Business Info, Properties, Documents, and Monetization. The main content area displays the GraphQL schema definition. A "Download Definition" button is located at the top right of the schema editor. The schema code is as follows:

```
1  enum MutationType {
2    CREATED
3    UPDATED
4    DELETED
5  }
6
7  # Meta information about the query.
8  type QueryMeta {
9    count: Int!
10 }
11
12 type AddToFilePlanetsPayload {
13   fileFileId: File!
14   planetFileId: Planet!
15 }
16
17 type AddToSpeciesPayload {
18   fileFileId: File!
19   speciesSpecies: Species!
20 }
21
22 type AddToStarshipsPayload {
23   fileFileId: File!
24   starshipStarship: Starship!
25 }
26
27 type AddToVehiclesPayload {
28   fileFileId: File!
29   vehicleVehicle: Vehicle!
30 }
31
32 type AddToPeopleFilePayload {
33   characterPerson: Person!
34   fileFileId: File!
35 }
36
37 type AddToPeoplePlanetsPayload {
38   residencePerson: Person!
39   knownPerson: Planet!
40 }
41
42 type AddToPeopleSpeciesPayload {
43   speciesSpecies: Species!
44 }
45
46 type AddToPeopleStarshipsPayload {
47   pilotPerson: Person!
48   starshipStarship: Starship!
49 }
```

GraphQL API from SDL

The screenshot shows the WSO2 API Manager interface for managing APIs. The left sidebar contains navigation links for Overview, Design Configurations, Runtime Configurations, Operations (which is selected and highlighted with a red box), Endpoints, Subscriptions, Lifecycle, Scheme Definition, Environments, Local Scopes, Business Info, Properties, Documents, and Monetization.

The main content area displays the configuration for the API "StarWarsAPI:1.0.0". The API is in the "CREATED" state. The "Operations" tab is active. Under "Operations Configuration", the "Rate limiting level" is set to "Operation Level" (highlighted with a red box). The "Operation" table lists five operations: Asset, Asset, Film, Film, Person, and Person. Each operation has its "Operation Type" (Asset, Film, or Person) and "Rate Limiting" (Unlimited) displayed. The "Scope" column contains dropdown menus for selecting an "Operation scope" (highlighted with a red box). The "Security Enabled" column shows toggle switches for each operation, all of which are turned on.

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

The screenshot shows the WSO2 API Manager interface. At the top, there's a navigation bar with tabs for PUBLISHER, APIs, Scopes, and API Products. Below the navigation bar, there's a sidebar titled "APIs" which says "Displaying 1 API(s)" and shows a thumbnail for "PizzaShackAPI". To the right of the sidebar, there's a "CREATE API ▾" button. A large callout box highlights the "Design New WebSocket API" option, which is described as "Design and prototype a new WebSocket API". Other options shown include "Design a New REST API", "I Have an Existing REST API", "I Have a SOAP Endpoint", and "I Have a GraphQL SDL schema". The bottom right corner of the interface has a small circular badge with the number "42".

PUBLISHER | APIs | Scopes | API Products

APIs
Displaying 1 API(s)

CREATE API ▾

Design a New REST API
Design and prototype a new REST API

I Have an Existing REST API
Use an existing OpenAPI definition file or URL

I Have a SOAP Endpoint
Use an existing SOAP or import the WSDL

I Have a GraphQL SDL schema
Import a GraphQL SDL schema

Design New WebSocket API
Design and prototype a new WebSocket API

PUBLISHED

42

WebSocket API

Create WebSocket API

Create a WebSocket API

Create a WebSocket API by providing a Name, a Context, and Business Plans (optional).

Name*
EchoWebSocket

Context*
/echowebsocket

Version*
1.0.0

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

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

Business plan(s)
Silver, Gold

Select one or more throttling policies for the API

* Mandatory fields

CREATE

CREATE & PUBLISH

CANCEL



WebSocket API

Overview WebSocket API

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

Header: WSO2 API MANAGER, APIs, Scopes, API Products, Search, SETTINGS, ADMIN.

API Overview: EchoWebSocket:1.0.0, CREATED State, Created by: admin.

Workflow Progress: Step 1: Created (green checkmark), Step 2: Endpoint (green checkmark), Step 3: Publish (blue bar).

Metadata:

Key	Value
Description	-
Provider	admin
Context	/echowebsocket
Version	1.0.0
Type	WS
Created Time	A few seconds ago
Last Updated Time	A few seconds ago
Business Owner	-
Technical Owner	-

Configuration:

Key	Value
Transports	(empty)
API Security	OAuth2
Access Control	None
Workflow Status	-
Visibility on Developer Portal	Public
Business Plans	Silver, Gold
Tags	-

Endpoints:

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

WebSocket API

Configuring endpoint

The screenshot shows the WSO2 API Manager interface. In the top navigation bar, there are links for APIS, Scopes, API Products, and a search bar. On the right, there are buttons for SETTINGS, ADMIN, Go To, Create New Version, Download API, and Delete. The main content area is titled "Endpoints". It lists two endpoints: "Production Endpoint" and "Sandbox Endpoint". Both endpoints have their URLs set to "ws://echo.websocket.org:80". There are edit icons next to each endpoint entry. At the bottom left, there are "Save" and "Cancel" buttons.

API with AWS Lambda Endpoint

The screenshot shows the WSO2 API Manager interface. On the left, there's a sidebar with various navigation options like Overview, Design Configurations, Runtime Configurations, Resources, Endpoints (which is selected), Subscriptions, Lifecycle, API Definition, Environments, Local Scopes, and Business Info. The main area displays an API named "PizzaShackAPI : 1.0.0" which is "PUBLISHED". It has a state of "State". Below this, under "Endpoints", there are four options: "HTTP/REST Endpoint", "HTTP/SOAP Endpoint", "Dynamic Endpoints", and "AWS Lambda" (which is highlighted with a red box). Under "Access Method", there are two options: "Using IAM role-supported temporary AWS credentials" (which is selected and highlighted with a blue circle) and "Using stored AWS credentials". There are input fields for "Access Key*", "Secret Key*", and "Region*". At the bottom, there are "Save" and "Cancel" buttons.

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

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

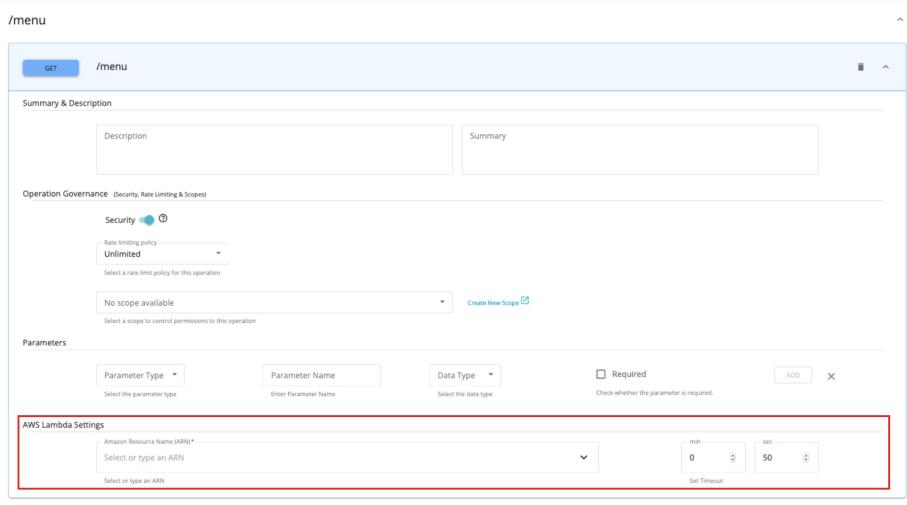
In AWS Lambda type, the credentials of the AWS API should be provided in this page.

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 API configuration interface. At the top, there's a navigation bar with a blue 'GET' button and the path '/menu'. Below it is a 'Summary & Description' section with 'Description' and 'Summary' fields. Under 'Operation Governance (Security, Rate Limiting & Scopes)', there's a 'Security' tab selected, showing a 'Rate limiting policy' dropdown set to 'Unlimited'. A note says 'Select a rate limit policy for this operation'. Below this is a dropdown menu with 'No scope available' and a 'Create New Scope' button. Under 'Parameters', there are fields for 'Parameter Type', 'Parameter Name', 'Data Type', and a 'Required' checkbox. A red box highlights the 'AWS Lambda Settings' section, which includes an 'Amazon Resource Name (ARN)' dropdown with 'Select or type an ARN' placeholder, and timeout fields for 'min' (0) and 'sec' (50). A 'Set Timeout' button is also present.

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.

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
- API Versioning
- Managing API Policies



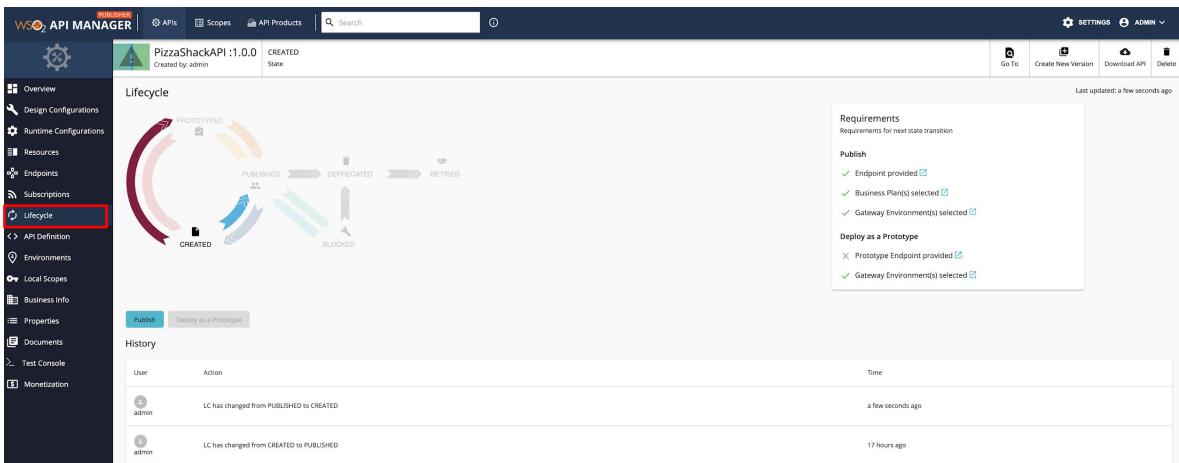
API Lifecycle

- States which represent the stages that an API has in the process of starting to develop an API until it's retirement.
 - Created
 - Prototyped
 - 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.

If the API is in CREATED state, it can either be PUBLISHED or PROTOTYPED.

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

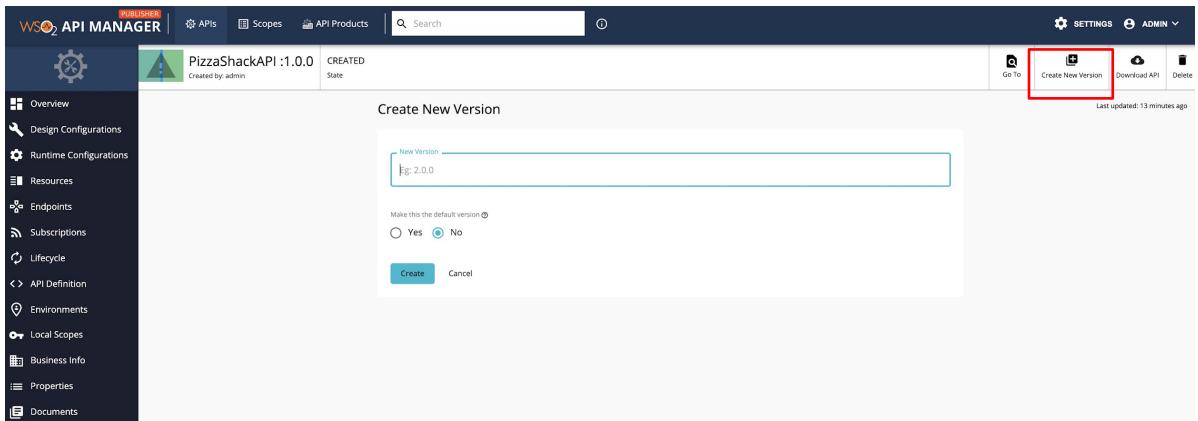
From CREATED state to PUBLISHED state, there must be a subscription policy applied as well as Endpoints should be provided.

To PROTOTYPES state, prototype endpoints should have to be provided.

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

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

The screenshot shows the WSO2 API Manager interface for managing the PizzaShackAPI version 2.0.0. The left sidebar contains navigation links for Overview, Design Configurations, Runtime Configurations, Resources, Endpoints, Subscriptions, Lifecycle, API Definition, Environments, Local Scopes, Business Info, Properties, Documents, Test Console, and Monetization. The main content area displays the Lifecycle of the API, which includes states: PROTOTYPED, PUBLISHED, DEPRECATED, and RETIRED. Below the Lifecycle diagram are two checkboxes: "Deprecate old versions after publishing the API" and "Requires re-subscription when publishing the API". A red box highlights these checkboxes. To the right of the Lifecycle diagram is a sidebar with sections for Requirements, Publish, Deploy as a Prototype, and Local Scopes. The Requirements section lists requirements for the next state transition. The Publish section shows successful validation steps: Endpoint provided, Business Plan(s) selected, and Gateway Environment(s) selected. The Deploy as a Prototype section shows a failed step: Prototype Endpoint provided. The Local Scopes section shows a successful step: Gateway Environment(s) selected.

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.

The screenshot shows the API management interface for the PizzaShackAPI 2.0.0. On the left, under 'Operations Configuration', the 'Rate limiting level' is set to 'API Level'. A dropdown menu for 'HTTP Verb' is open. On the right, for the '/menu' resource, a modal window displays four throttling policy options: '10KPerMin', '20KPerMin', '50KPerMin', and 'Unlimited'. The 'Unlimited' option is selected. The modal also includes fields for 'Description' (containing '10KPerMin', '20KPerMin', '50KPerMin', and 'Unlimited') and 'Operation General' (containing '10KPerMin', '20KPerMin', '50KPerMin', and 'Unlimited'). Below the modal, there is a note: 'No scope available. Select a scope to control permissions to this operation.' At the bottom of the modal, there are 'Parameter Type' and 'Parameter Name' fields.

Throttling policies can be added per API or per resource.

More on Throttling policies in the Throttling section

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

The screenshot shows the WSO2 API Manager Publisher Portal. In the top navigation bar, the 'PUBLISHER' tab is selected. The main content area displays the 'Test Console' for the 'PizzaShackAPI : 1.0.0' API, which is currently in the 'CREATED' state. A red box highlights the 'INITIALIZE TEST' button. On the left side, the navigation menu is visible, with the 'Test Console' item also highlighted by a red box.

Testing APIs is trying out the APIs at the publisher itself to make sure whether the functionalities and behaviours are met before publishing to the gateway for subscribers to access

Only the API developers (creator, publisher) are allowed to test the APIs through the test console.

The developers can do the basic functional tests such as checking mediation policy, validating that the request is sent to the back-end and a proper responses is received as expected, request/response schema validation, and making sure that the API resources are defined correctly.

The Test Console menu item in the Publisher portal's left navigation menu can be used to initialize the test at the design phase. Once the developer initiates the test by clicking on the initialize test button, the API is transformed to the prototype(testing) state and the swagger console is populated with a test-key(token) which prevents the unauthorized users accessing the particular API.

API must be in the **created** lifecycle state, the developer can initialize the test.

Testing APIs

After initiating tests, the swagger UI(API Console) to test the API will be available

The screenshot shows the API management interface for the PizzaShackAPI version 1.0.0. At the top, there's a navigation bar with icons for search, view in Dev Portal, create new version, download API, and delete. Below the navigation is a status bar indicating the API is prototyped and last updated a few seconds ago. The main area is titled "Test Console" and contains a "Gateway" section. In the "Gateway" section, there's a dropdown menu for "Environment" set to "Production and Sandbox". Below it, a "Schemes" dropdown is set to "HTTPS". Under the "default" section, there are five API endpoints listed: POST /order (green), GET /menu (blue), GET /order/{orderId} (light blue), PUT /order/{orderId} (orange), and DELETE /order/{orderId} (red). Each endpoint has a lock icon to its right.

Click on the **Initiate Test** button. This will open the swagger UI(API Console) to test the PizzaShack API before publish to the subscribers.

Testing APIs

Test the resource

The screenshot shows a Swagger UI interface for a POST method to the '/order' endpoint. The 'body' field is marked as required and contains the following JSON schema:

```
{ "customerName": "string", "deliveryAddress": "string", "address": "string", "pizzaType": "string", "creditCardNumber": "string", "quantity": 0, "orderId": "string" }
```

The 'Parameter content type' dropdown is set to 'application/json'. Below the form, there are sections for 'Responses' and 'Curl'. The 'Curl' section contains a generated command:

```
curl -X POST "https://localhost:9443/order" -H "accept: application/json" -H "Content-Type: application/json" -H "test-key: d088575f-e71d-4446-91cf-b4f7abb01098" -d '{"customerName": "string", "deliveryAddress": true, "address": "string", "pizzaType": "string", "creditCardNumber": "string", "quantity": 0, "orderId": "string"}'
```

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



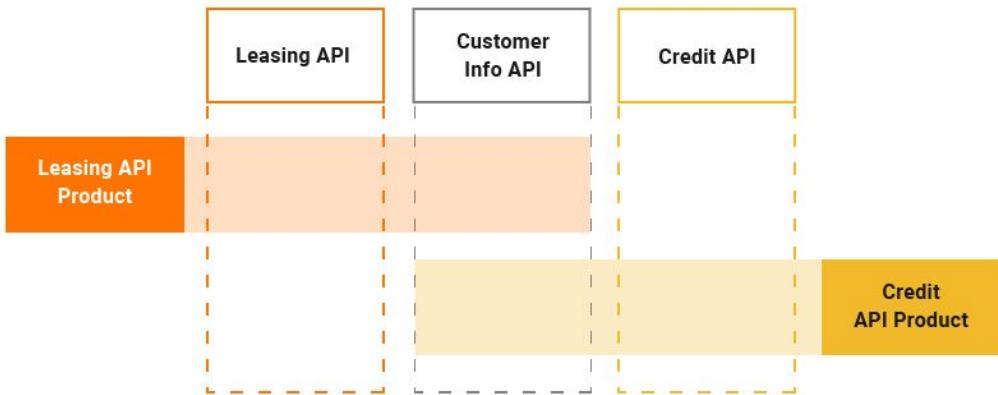
Let's try it out!

Creating and publishing an API Product with API Manager Publisher



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.



Publishing an API Product

Define API Product

The screenshot shows the WSO2 API Manager interface with the 'PUBLISHER' tab selected. In the top navigation bar, the 'API Products' menu item is highlighted with a red box. Below the navigation, a search bar and settings/publisher links are visible. The main content area is titled 'Create an API Product' with the sub-instruction 'Create an API Product by providing a Name, a Context, Resources, and Business Plans (optional).'. The form fields include:

- Name***: customer-leasing
- Context***: customer-leasing
- Business plan(s)***: Bronze (selected)

Below the form, a note states: 'API Product will be exposed in customer-leasing(version) context at the gateway'. A 'Select one or more throttling policies for the API Product' dropdown is shown with a placeholder 'None'. A small note indicates '* Mandatory fields'. At the bottom are 'CANCEL' and 'NEXT' buttons.

- 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

The screenshot shows the WSO2 API Manager interface for creating a new API product. On the left, there's a sidebar with a search bar and a 'PUBLISHER' tab. The main area has a title 'Create an API Product' and a subtitle 'Create an API Product by providing a Name, a Context, Resources, and Business Plans (optional)'. It features three panels: 'Select an API' (listing 'customer-info' and 'leasing'), 'Select API Resources' (listing resources for 'customer-info' like '/customers', '/customers/{customerId}', etc.), and 'Add Resources' (showing the selected resources for the product 'customer-leasing'). At the bottom are 'BACK' and 'PUBLISH' buttons.

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 WSO2 API Manager interface with the following details:

Header: WSO2 API MANAGER | APIs | API Products | Search | BACK TO PRODUCTS | customer-leasing | PUBLISHED | SETTINGS | PUBLISHER | Go To | View in Dev Portal | Delete | Last updated: a few seconds ago

Left Sidebar: Overview, Design Configurations, Runtime Configurations, Resources, Subscriptions, API Definition, Business Info, Properties, Documents, Monetization.

Overview Section:

- Metadata:**
 - Description: publisher
 - Context: /customer-leasing
 - Created Time: A few seconds ago
 - Last Updated Time: A few seconds ago
 - Business Owner: -
 - Technical Owner: -
- Configuration:**
 - Transports: HTTP, HTTPS
 - API Security: OAuth2
 - Access Control: None
 - Workflow Status: -
 - Visibility on Developer Portal: Public
 - Business Plans: Bronze
- Resources:**
 - customer-info
 - /customers [edit]
 - /customers/{customerId} [edit]
 - leasing
 - /assets [edit]
 - /assets/{assetId} [edit]

Important to note:

- API Products can be only created by users with **publisher** role.
- The only state API Products could have is the **PUBLISHED** state.
- 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.