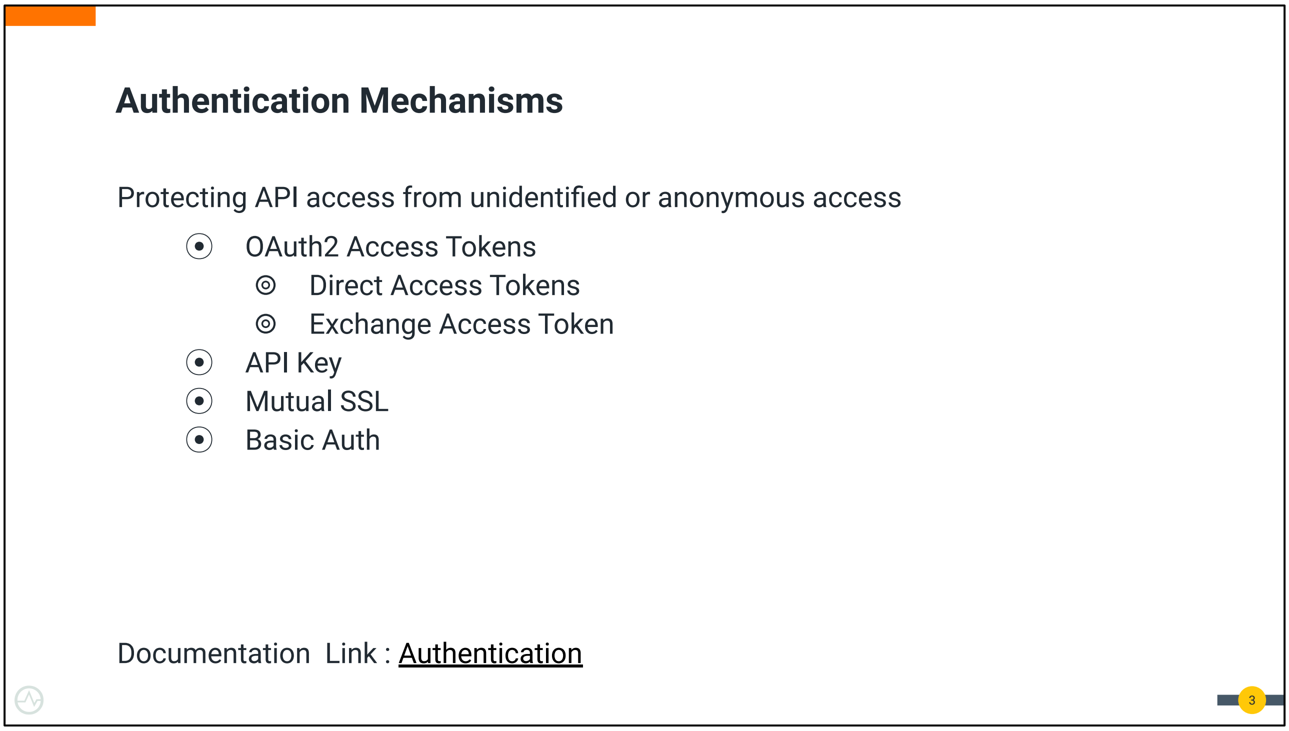
WSO2 API Manager 4.2.0 Developer Fundamentals

API Security

WSO2 Training

Content



Authentication Mechanisms

Protecting API access from unidentified or anonymous access

- OAuth2 Access Tokens
  - Direct Access Tokens
  - Exchange Access Token
- API Key
- Mutual SSL
- Basic Auth

Documentation Link : Authentication

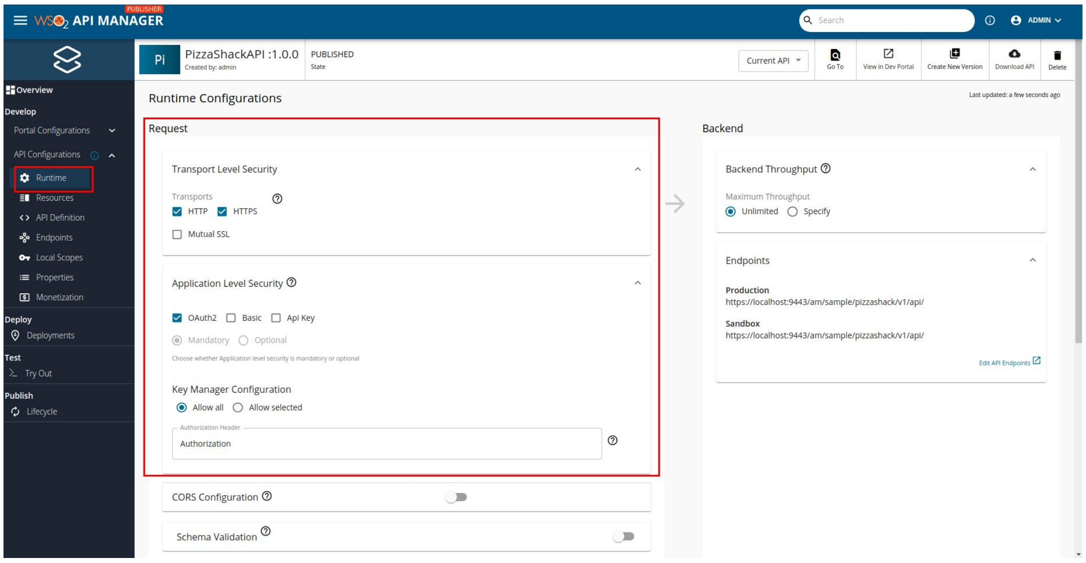
Authentication: Verification of the identity of the user/ party who is accessing the particular resource.

API Manager supports multiple types of API Authentication mechanisms. It is also possible to combine them as required through the Publisher portal.

E.g.,

- Configure only OAuth2/ API Key or Basic auth
- Configure Mutual SSL + Oauth2 (Both mechanisms are mandatory)
- Configure Mutual SSL + OAuth2 (either one mechanism is required)

Selecting Authentication Mechanism for API



Pi PizzaShackAPI :1.0.0
Created by: admin
PUBLISHED
State

Runtime Configurations

Last updated: a few seconds ago

Request

Transport Level Security

Transports
[✓ HTTP] [✓ HTTPS]
Mutual SSL

Application Level Security

[✓ OAuth2] Basic Api Key
Mandatory Optional
Choose whether Application level security is mandatory or optional

Key Manager Configuration
[✓ Allow all] Allow selected

Authorization Header: Authorization

CORS Configuration

Schema Validation

Backend

Backend Throughput

Maximum Throughput
[✓ Unlimited] Specify

Endpoints

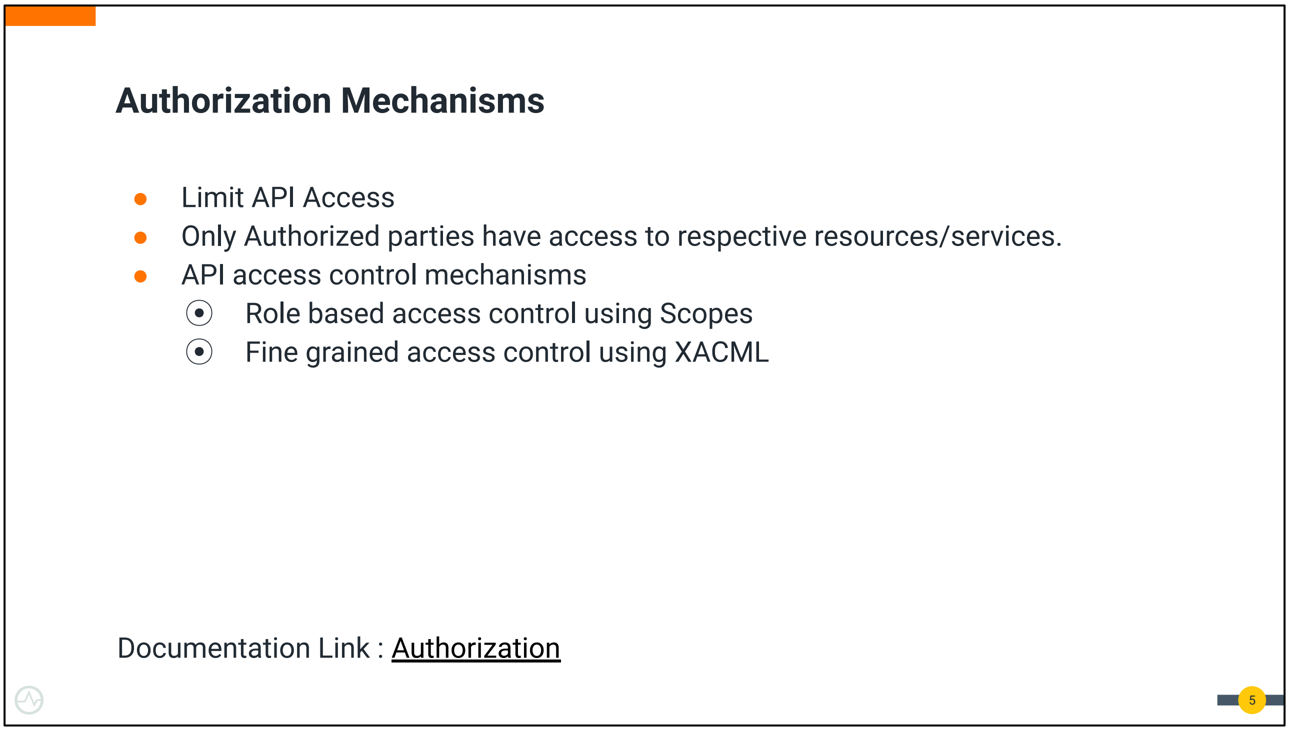
Production
https://localhost:9443/am/sample/pizzashack/v1/api/

Sandbox
https://localhost:9443/am/sample/pizzashack/v1/api/

Edit API Endpoints

Log into API Publisher, and select an API.
Go to the API Configurations page.
Under Runtime, expand the **Transport Level Security** and the **Application Level Security** in the Request section.

- Select the Required Application level security.
- Enable Mutual SSL if required and upload client certificate
- Change the default Authorization header name.



Authorization Mechanisms

- Limit API Access
- Only Authorized parties have access to respective resources/services.
- API access control mechanisms
  - ⦿ Role based access control using Scopes
  - ⦿ Fine grained access control using XACML

Documentation Link : Authorization

Authorization: Verifying whether the authenticated entity is permitted to access the particular resource.

Role-based access control using scopes
API Resources can be secured with scopes which are bound to specific user roles.

E.g,: In an Educational Institute, the lecturer role can add course content, admin-staff role can register new courses and students, student role can only get the course details.

Scope: course-content:add | **Associated Roles:** lecturer
Scope: course:create | **Associated Roles**: admin-staff
Scope: course:get | **Associated Roles**: student, lecturer, admin-staff

Applying scopes to resources.

POST	/course/content	-> scope: course-content:add
POST	/course	-> scope: course:create
GET	/course/{id}	-> scope: course:get

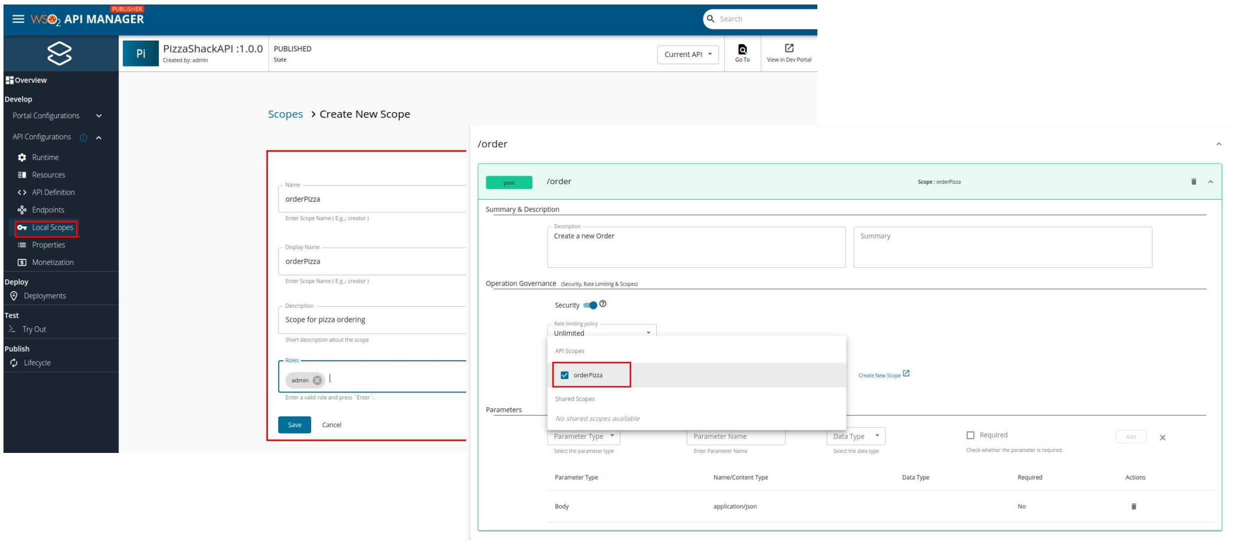
Fine-grained access control using XACML (eXtensible Access Control Markup Language)
API Resources are secured with XACML policies. This mechanism can be used when IS as key manager is configured with API Manager.

In this mechanism, other user attributes such as Organization or any custom claim can be used to authorize the API request.

E.g.,:

- Allowing only the users who belong to the accounts department to access account resources
- Allow only the students that are registered to specific course codes to access those course details
- Deny any API Request from users who are under aged.

Creating Local Scopes and Assigning to Resources



Name: orderPizza
Display Name: orderPizza
Description: Scope for pizza ordering
Roles: admin

/order
post /order — Scope : orderPizza
Description: Create a new Order
Summary:
Rate limiting policy: Unlimited
API Scopes: ✓ orderPizza
Shared Scopes: No shared scopes available

Parameter Type	Name/Content Type	Data Type	Required	Actions
Body	application/json		No	

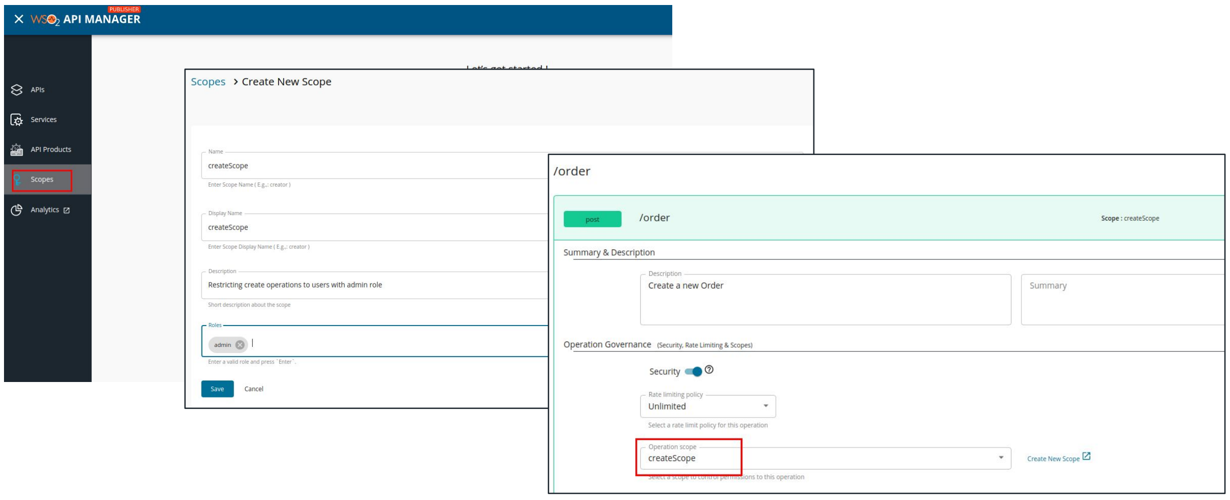
Create a Local Scope

- Login to API Publisher and open an API by selecting it.
- Go to API configurations -> Local Scopes page from the left menu.
- Click on Create Scopes button
- Provide the name for the scope, display name, description and associated roles list.
- Click Save to create the scope.

Assign Scope to Resources

- Go to API Configurations -> Resources page.
- Expand the required resource.
- Select the created scope from the drop down list
- Save the changes

Creating Shared Scopes and Assigning to Resources



Name: createScope
Display Name: createScope
Description: Restricting create operations to users with admin role
Roles: admin

/order
post /order — Scope : createScope
Summary & Description
Description: Create a new Order
Summary
Operation Governance (Security, Rate Limiting & Scopes)
Security: on
Rate limiting policy: Unlimited
Operation scope: createScope

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

Create a Scope

- Login to API Publisher.
- Go to Scopes page from the side menu
- Click on Scopes button
- Provide the name for the scope, description and associated roles list.
- Click Save to create the scope.

Assign Scope to Resources

- Go to API Configurations -> Resources page.
- Expand the required resource.
- Select the created scope from the drop down list
- Save the changes

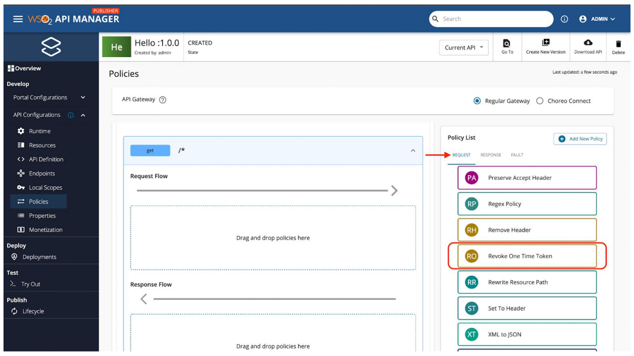
Bot Detection

- Hackers can invoke the APIs without an access token by scanning the open ports of a system
- Bot detection mechanism
  - Traces and logs details of such unauthorized API calls in the <API-M_HOME>/repository/logs/wso2-BotDetectedData.log file.

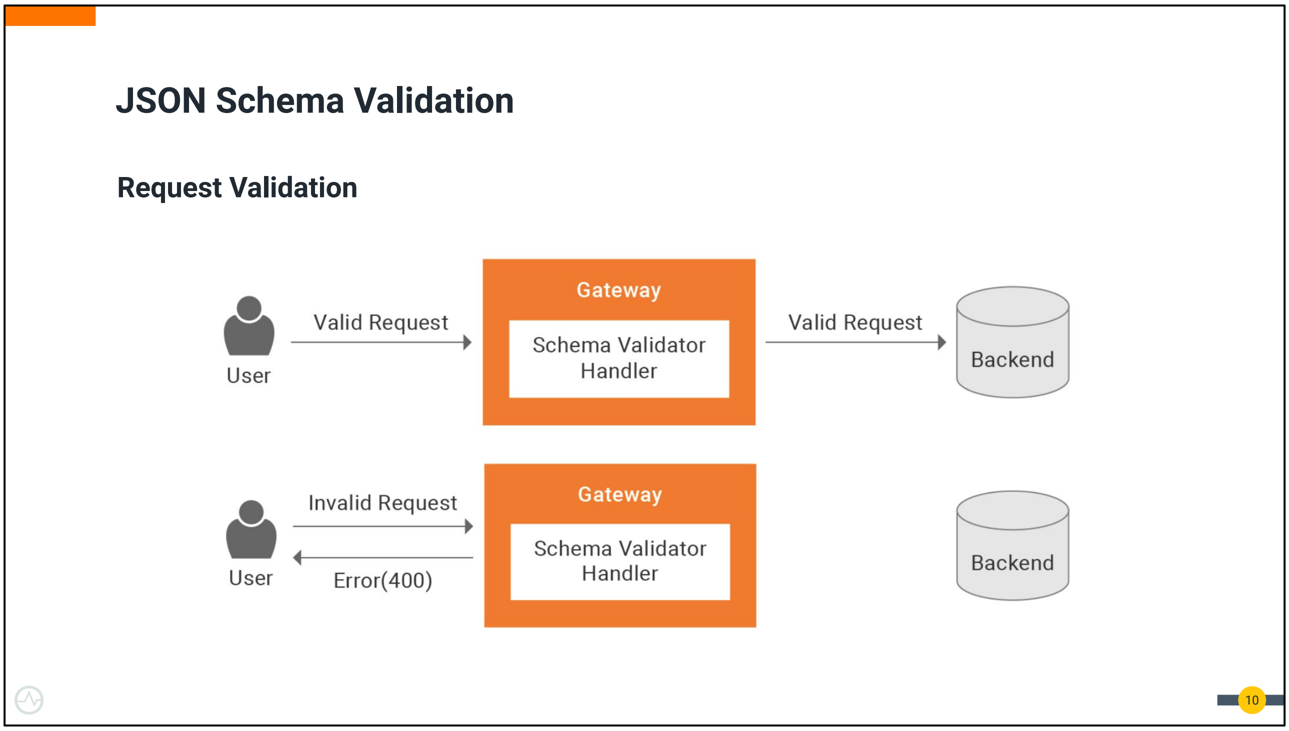
Documentation Link: Bot Detection

One time token policy

- One Time Token is a JWT that will be revoked after single usage
- Enabled by attaching a policy to the resource

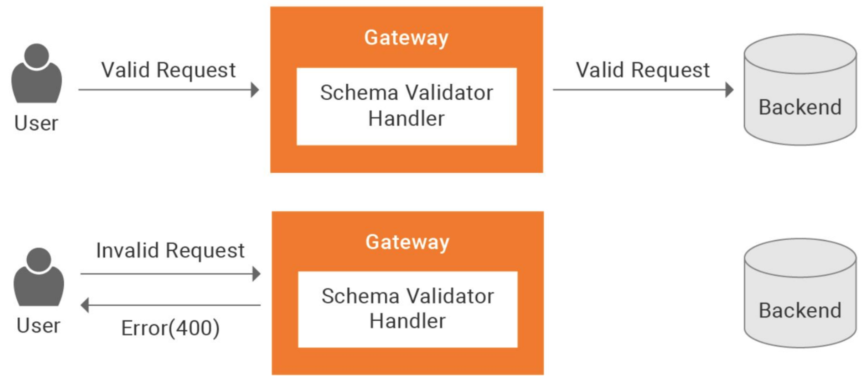


Documentation Link: Revoke One Time Tokens Policy



JSON Schema Validation

Request Validation



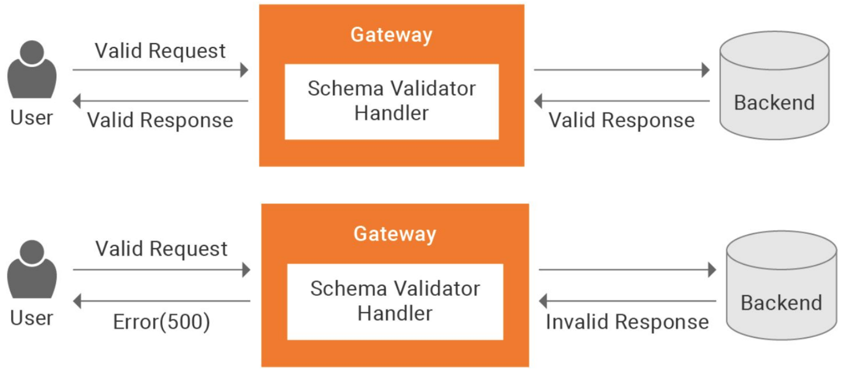
When creating APIs, it is possible to enable request/ response schema validation for API requests.

If enabled, the gateway will validate the request payload against the payload format defined in the swagger definition. If the payload did not match, (data types, missing required fields) gateway will return a 400 - Bad Request response back to the client.

If the validation is successful the request will be passed to the backend.

JSON Schema Validation

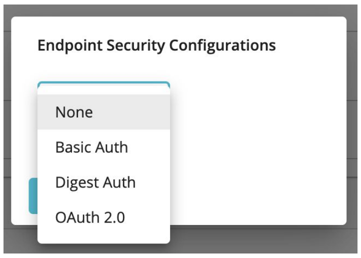
Response Validation



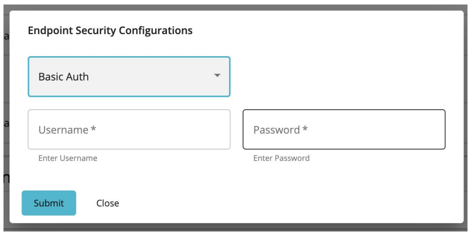
In the Response validation, if the returning response from the backend service does not match with the expected response object, a 500 response will be sent to the client.

Using Secured Endpoints

- Basic Auth
- Digest Auth
- OAuth 2.0


Endpoint Security Configurations

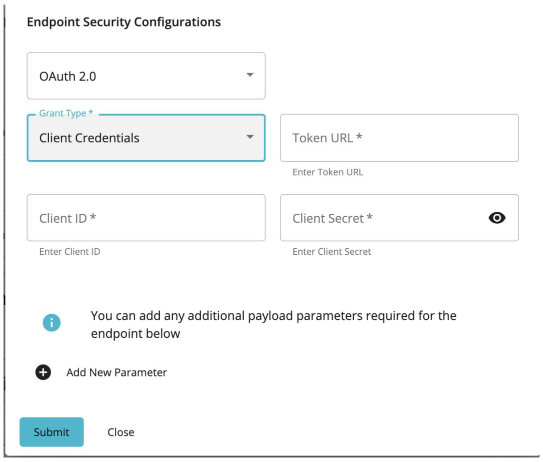
- None
- Basic Auth
- Digest Auth
- OAuth 2.0


Endpoint Security Configurations

Basic Auth

Username * — Enter Username

Password * — Enter Password


Endpoint Security Configurations

OAuth 2.0

Grant Type *: Client Credentials

Token URL * — Enter Token URL

Client ID * — Enter Client ID

Client Secret * — Enter Client Secret

You can add any additional payload parameters required for the endpoint below

Documentation Link: Secure Endpoints

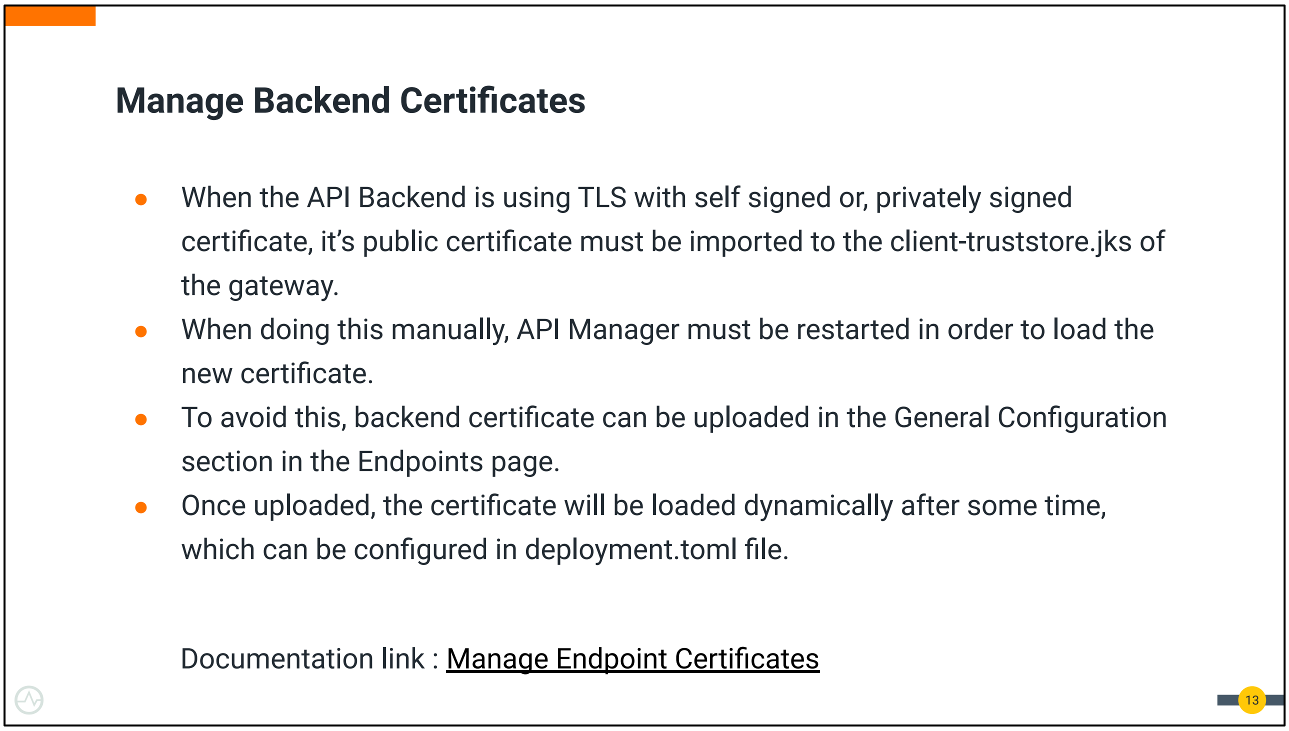
When the back-end endpoints are secured with basic, digest or OAuth 2.0 authentication, it can be configured when adding the endpoints to the API.

These endpoint credentials can also be encrypted by enabling secure vault in <APIM_HOME>/repository/conf/deployment.toml file.
https://apim.docs.wso2.com/en/4.2.0/install-and-setup/setup/security/logins-and-passwords/working-with-encrypted-passwords/#encrypting-secured-endpoint-passwords

OAuth 2.0

OAuth brings in a separate authorization layer in order to separate the role of the client from that of the resource owner. In order to gain access to a protected resource, the client should obtain a set of properties including an access token, its lifetime, and scope instead of credentials of the resource owner, from the backend server.
The OAuth 2.0 Authorization Framework specification declares 4 grant-types to obtain an access token. OAuth 2.0 Endpoint Security supports the following grant types:

1. Client Credentials
2. Resource Owner Password



Manage Backend Certificates

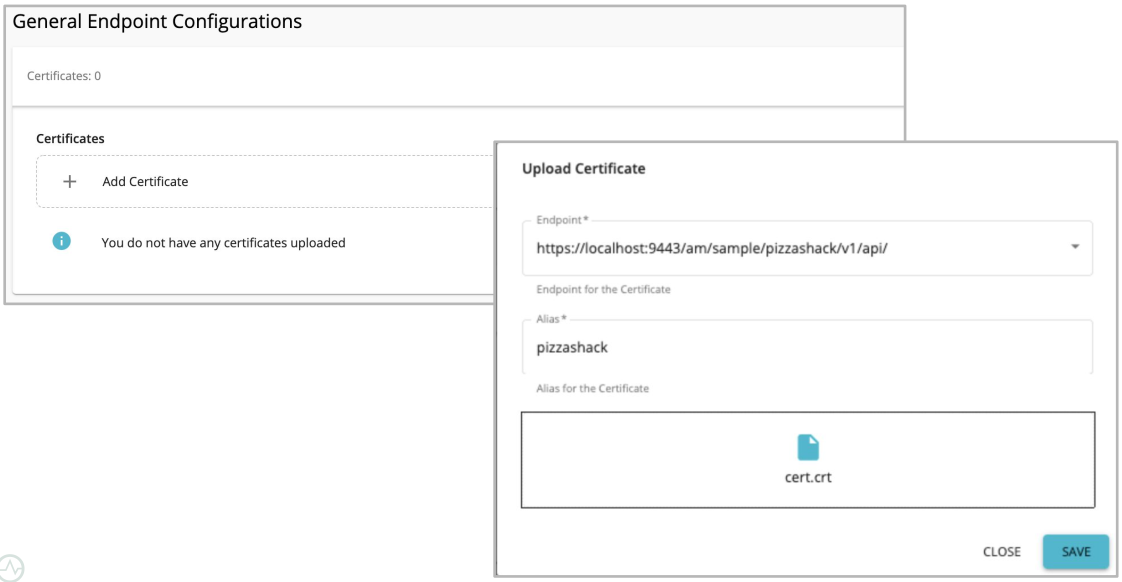
- When the API Backend is using TLS with self signed or, privately signed certificate, it's public certificate must be imported to the client-truststore.jks of the gateway.
- When doing this manually, API Manager must be restarted in order to load the new certificate.
- To avoid this, backend certificate can be uploaded in the General Configuration section in the Endpoints page.
- Once uploaded, the certificate will be loaded dynamically after some time, which can be configured in deployment.toml file.

Documentation link : Manage Endpoint Certificates

Configuring the certificate loading interval.

```
[transport.passthru_https.sender.ssl_profile]
interval = 600000
```

Upload Backend Certificates


General Endpoint Configurations
Certificates: 0
Certificates
Add Certificate
You do not have any certificates uploaded
Upload Certificate
CLOSE
SAVE

Endpoint*: https://localhost:9443/am/sample/pizzashack/v1/api/

Endpoint for the Certificate

Alias*: pizzashack

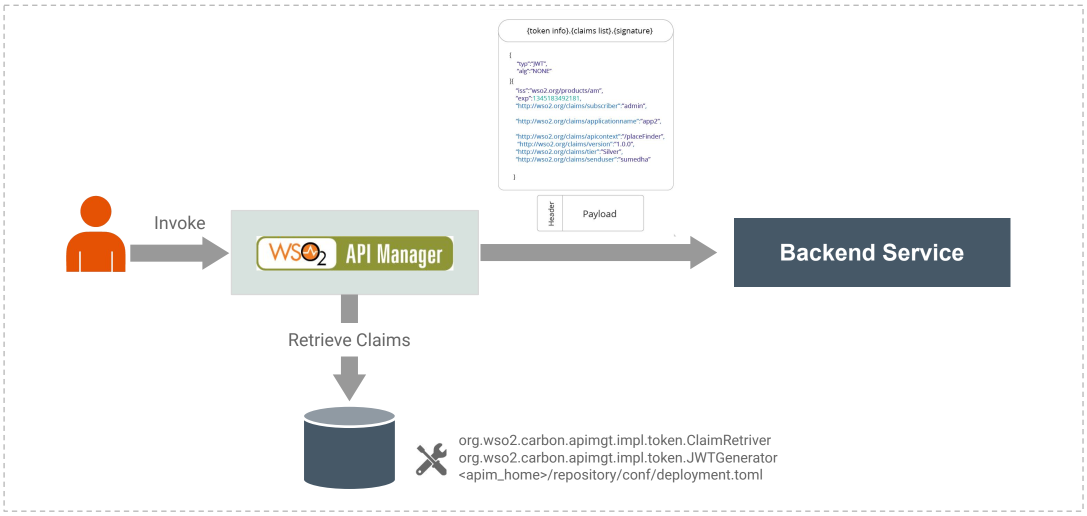
Alias for the Certificate

cert.crt

In the API Publisher portal,

- Click on an API and goto API configurations -> Endpoints
- Expand the General Endpoint Configuration
- In Certificate, click on Add Certificate
- In the Upload Certificate dialog box,
    - Select the Endpoint
    - Provide an Alias
    - Select the certificate file.
- Click on Save button to upload.

Passing End-User Attributes to Backend.



```
{token info}.{claims list}.{signature}

{
    "typ":"JWT",
    "alg":"NONE"
}{
   "iss":"wso2.org/products/am",
   "exp":1345183492181,
   "http://wso2.org/claims/subscriber":"admin",

   "http://wso2.org/claims/applicationname":"app2",

   "http://wso2.org/claims/apicontext":"/placeFinder",
    "http://wso2.org/claims/version":"1.0.0",
   "http://wso2.org/claims/tier":"Silver",
   "http://wso2.org/claims/senduser":"sumedha"

  }
```

```
org.wso2.carbon.apimgt.impl.token.ClaimRetriver
org.wso2.carbon.apimgt.impl.token.JWTGenerator
<apim_home>/repository/conf/deployment.toml
```

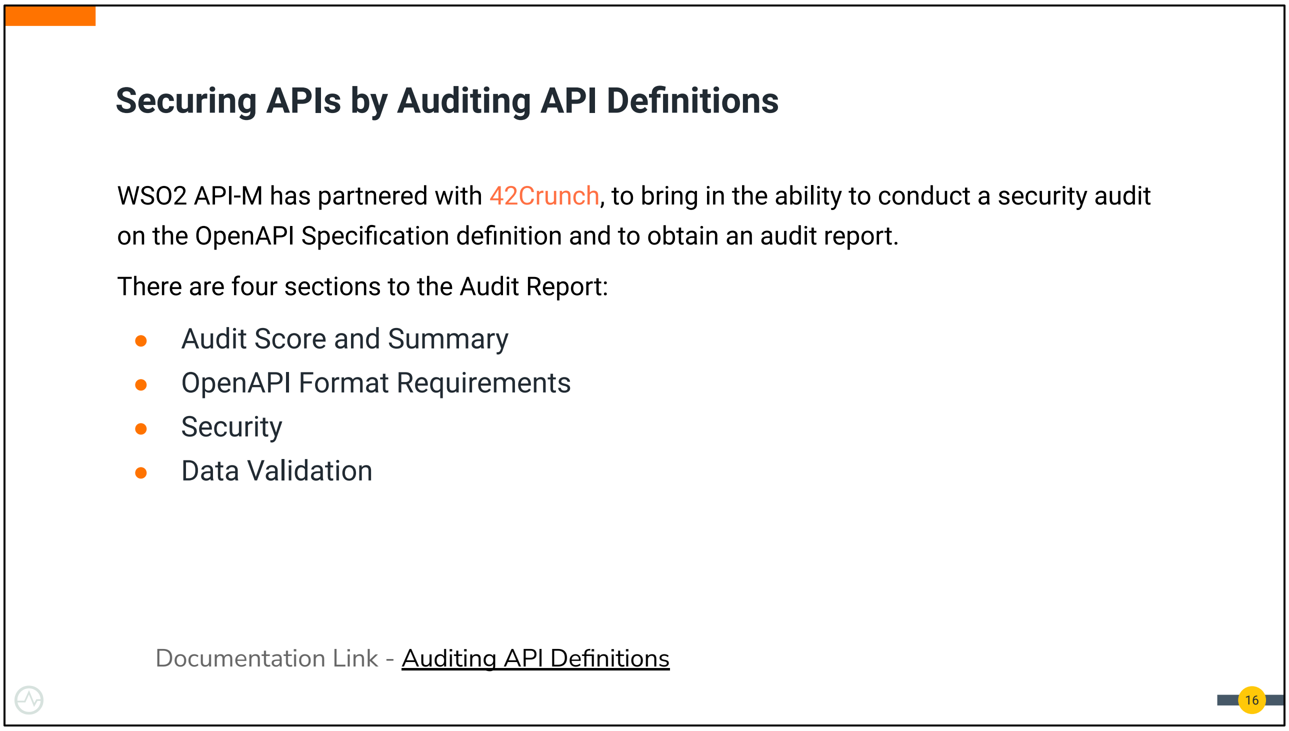
Documentation Link - Passing JWT to backend

JSON Web Token (JWT) is used to represent claims that are transferred between two parties such as the end user and the backend.

A claim is an attribute of the user that is mapped to the underlying user store. It is encoded as a JavaScript Object Notation (JSON) object that is used as the payload of a JSON Web Signature (JWS) structure, or as the plain text of a JSON Web Encryption (JWE) structure. This enables claims to be digitally signed.

When a request comes to the API Gateway, it retrieves the claims needed and builds a JWT token which is added to the header of the backend service request. The backend would get this and do the necessary validations.

The JWT that is generated by default has predefined attributes that are passed to the backend. These include basic application-specific details, subscription details, and user information that are defined in the JWT generation class that comes with the API Manager by the name org.wso2.carbon.apimgt.impl.token.JWTGenerator. If you want to pass additional attributes to the backend with the JWT or completely change the default JWT generation logic you can extend this process.



Securing APIs by Auditing API Definitions

WSO2 API-M has partnered with 42Crunch, to bring in the ability to conduct a security audit on the OpenAPI Specification definition and to obtain an audit report.

There are four sections to the Audit Report:

- Audit Score and Summary
- OpenAPI Format Requirements
- Security
- Data Validation

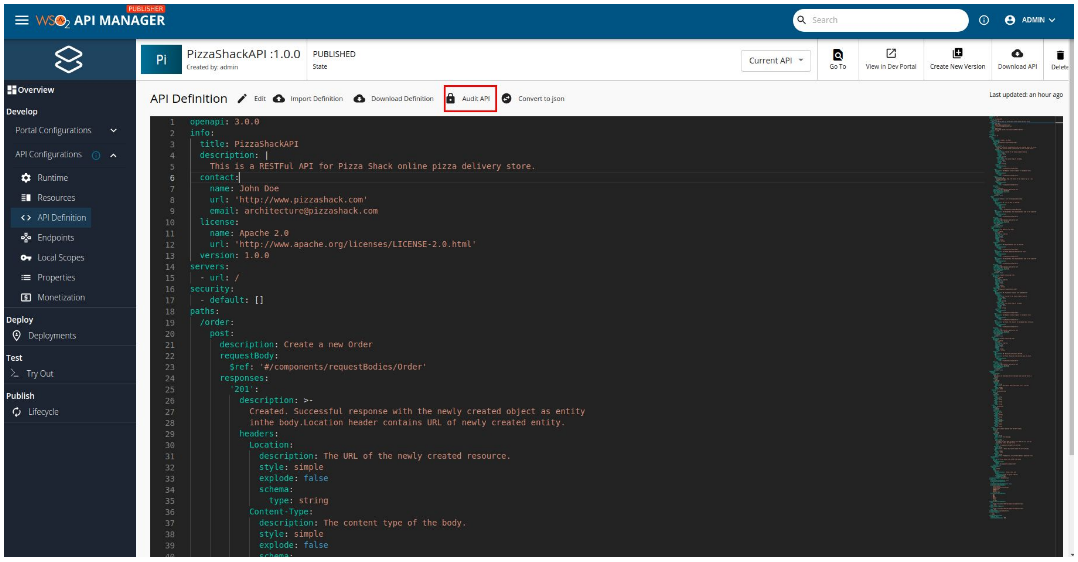
Documentation Link - Auditing API Definitions

API Security has become an important concern in recent times as organizations are more cautious about exposing raw, sensitive data via APIs. Therefore, it is important that APIs adhere to the OpenAPI Specification (OAS) to ensure API security.
WSO2 API-M has partnered with 42Crunch, the only enterprise API security platform, to bring in the ability to conduct a security audit on the OpenAPI Specification definition and to obtain an audit report.

Audit report sections
There are four sections to the Audit Report:

1. Audit Score and Summary
2. OpenAPI Format Requirements
3. Security
4. Data Validation

Securing APIs by Auditing API Definitions



```yaml
openapi: 3.0.0
info:
  title: PizzaShackAPI
  description: |
    This is a RESTFul API for Pizza Shack online pizza delivery store.
  contact:
    name: John Doe
    url: 'http://www.pizzashack.com'
    email: architecture@pizzashack.com
  license:
    name: Apache 2.0
    url: 'http://www.apache.org/licenses/LICENSE-2.0.html'
  version: 1.0.0
servers:
  - url: /
security:
  - default: []
paths:
  /order:
    post:
      description: Create a new Order
      requestBody:
        $ref: '#/components/requestBodies/Order'
      responses:
        '201':
          description: >-
            Created. Successful response with the newly created object as entity
            inthe body.Location header contains URL of newly created entity.
          headers:
            Location:
              description: The URL of the newly created resource.
              style: simple
              explode: false
              schema:
                type: string
            Content-Type:
              description: The content type of the body.
              style: simple
              explode: false
              schema:
```

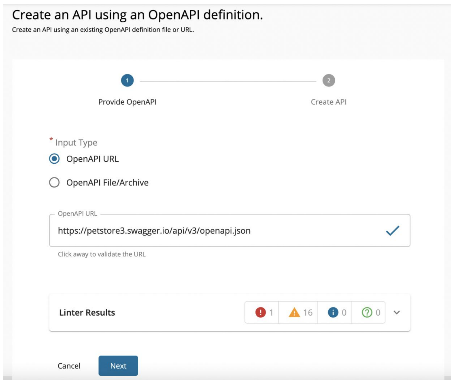
API Security has become an important concern in recent times as organizations are more cautious about exposing raw, sensitive data via APIs. Therefore, it is important that APIs adhere to the OpenAPI Specification (OAS) to ensure API security.
WSO2 API-M has partnered with 42Crunch, the only enterprise API security platform, to bring in the ability to conduct a security audit on the OpenAPI Specification definition and to obtain an audit report.

Refer
https://apim.docs.wso2.com/en/4.2.0/design/api-security/configuring-api-security-audit/#securing-apis-by-auditing-api-definitions

Audit report sections
There are four sections to the Audit Report:

1. Audit Score and Summary
2. OpenAPI Format Requirements
3. Security
4. Data Validation

Validating API Definitions Against Custom Linter Rules



Create an API using an OpenAPI definition.

Create an API using an existing OpenAPI definition file or URL.

* Input Type

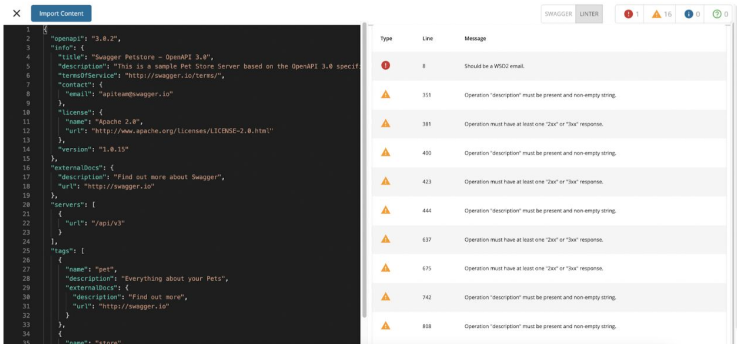
(●) OpenAPI URL

( ) OpenAPI File/Archive

OpenAPI URL: https://petstore3.swagger.io/api/v3/openapi.json ✓

Click away to validate the URL

Linter Results ● 1 ⚠ 16 ● 0 ? 0



```
1   {
2     "openapi": "3.0.2",
3     "info": {
4       "title": "Swagger Petstore - OpenAPI 3.0",
5       "description": "This is a sample Pet Store Server based on the OpenAPI 3.0 specif
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"
```

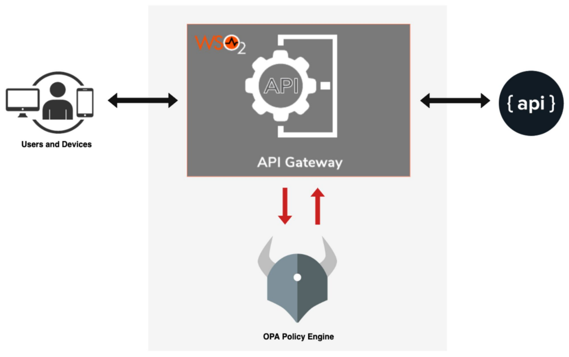
Type	Line	Message
●	8	Should be a WSO2 email.
⚠	351	Operation "description" must be present and non-empty string.
⚠	381	Operation must have at least one "2xx" or "3xx" response.
⚠	400	Operation "description" must be present and non-empty string.
⚠	423	Operation must have at least one "2xx" or "3xx" response.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

Documentation link : Validate definitions with linters

Open Policy Agent (OPA)

- The Open Policy Agent (OPA) is an open source, general-purpose policy engine that unifies policy enforcement.
- With the ability to define complex policies as a code, OPA provides much needed flexibility over standard static access control mechanisms such as role-based or attribute-based access control.
- Both WSO2 API Manager regular Gateway and Choreo Connect supports Open Policy Agent-based request validation

Open Policy Agent(OPA)


Users and Devices
WSO2
API
API Gateway
{ api }
OPA Policy Engine