



WSO2 API Manager 3.2.0 Developer Fundamentals

API Security



WSO2 Training

CC BY 4.0

Content

- Authentication Mechanisms
- Authorization Mechanisms
- Bot Detection
- JSON Schema Validation
 - ◉ Request Validation
 - ◉ Response Validation
- Using Secured Endpoints
- Passing End User Attributes to Backend.



Authentication Mechanisms

Protecting API access from unidentified or anonymous access

- OAuth2 Access Tokens
 - ⊙ Self Contained Access Tokens(JWT)
- 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

The screenshot displays the WS2 API Manager interface. The top navigation bar includes 'APIs', 'Scopes', and 'API Products'. The left sidebar lists various management options, with 'Runtime Configurations' highlighted. The main content area shows the 'Runtime Configurations' for 'PizzaShackAPI : 1.0.0'. The 'Request' section is expanded, showing 'Transport Level Security' with 'HTTP' and 'HTTPS' selected, and 'Application Level Security' with 'OAuth2' selected and 'Mandatory' chosen. The 'Backend' section shows 'Backend Throughput' set to 'Unlimited' and 'Endpoints' for 'Production' and 'Sandbox'.

Log into API Publisher, and select an API.

Go to the Runtime Configurations page.

In the Runtime Configurations, 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

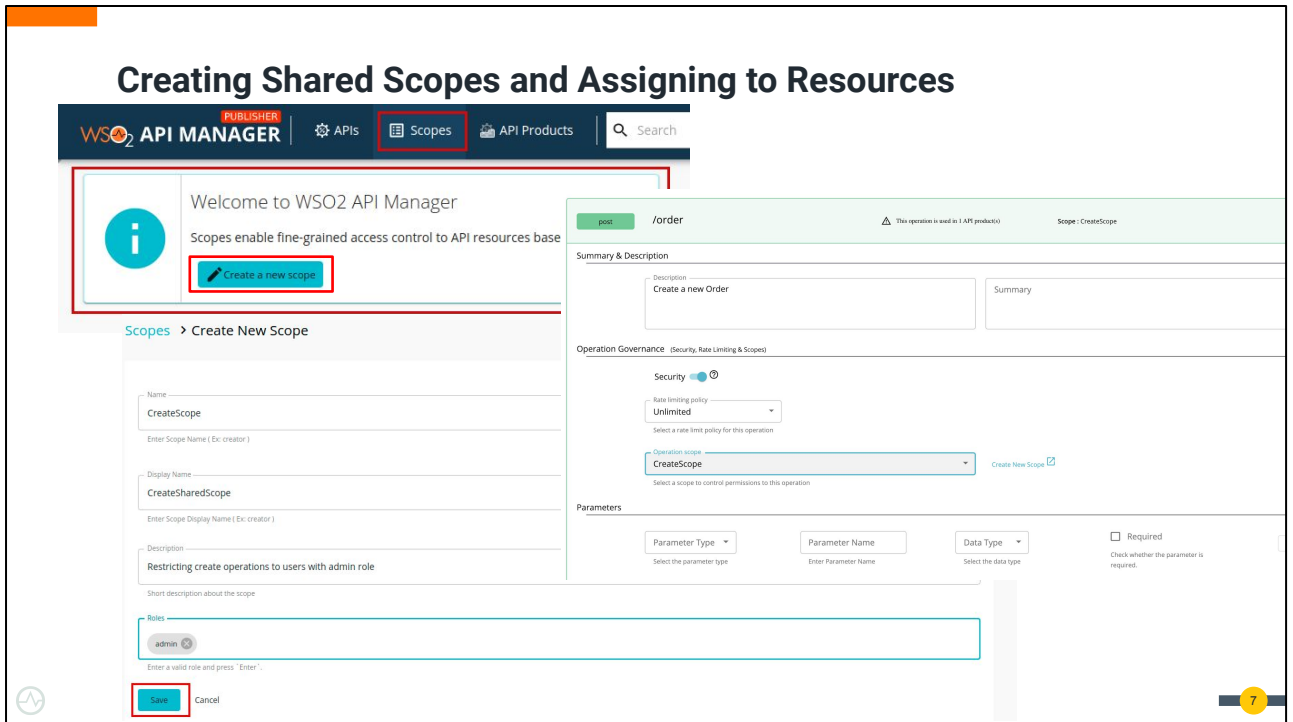
The screenshot displays two overlapping forms from the WSO2 API Manager interface. The background form is the 'Create New Scope' page for the 'PizzaShackAPI :1.0.0'. It includes fields for 'Name' (filled with 'orderPizza'), 'Description' (filled with 'Scope for Pizza ordering'), and 'Roles' (filled with 'admin'). The foreground form is the 'Assign Scope to Resources' page for the '/order' resource. It shows a 'Security' section with a dropdown menu for 'Operation Scope' where 'orderPizza' is selected. Both forms have 'Save' and 'Cancel' buttons.

Create a Local Scope

- Login to API Publisher and open an API by selecting it.
- Go to Local Scopes page from the left menu.
- Click on Create 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 Resources page.
- Expand the required resource.
- Select the created scope from the drop down list
- Save the changes



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 and open an API by selecting it.
- Go to Scopes page from the top menu bar
- Click on Create a New Scope button
- Provide the name for the scope, description and associated roles list.
- Click Save to create the scope.

Assign Scope to Resources

- Go to 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.
 - ⦿ Sends notifications in this regard via emails
 - ⦿ Bot detected data can be viewed via the Admin Portal (need to configure Analytics)

Documentation Link: [Bot Detection](#)



New Feature in APIM 3.2

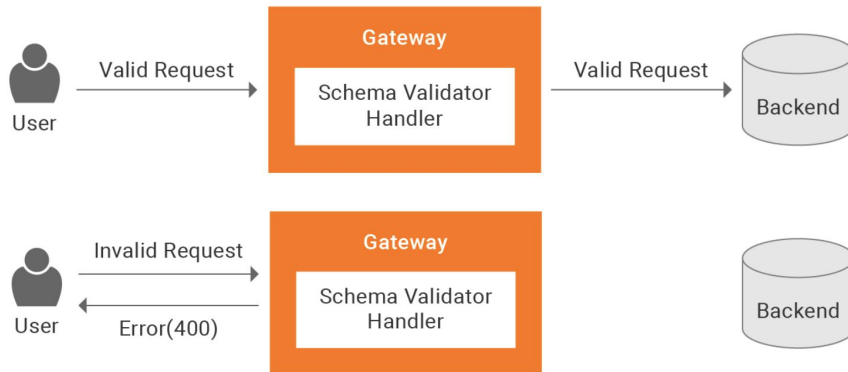
There is a possibility that hackers may invoke services without any proper authentication by using tools such as port scanning. Therefore, WSO2 API Manager (WSO2 API-M) provides a bot detection mechanism in place

There is an unadvertised service deployed in the gateway which logs and sends events to analytics if it receives any traffic. These events can then be configured to send e-mail alerts and also shown in the bot detection dashboard in the Admin Portal. Thereby this helps Publishers to protect their data from bot attackers and improve the security of the data.

If hackers (e.g., bot attackers) try to invoke the unadvertised service, WSO2 API Manager will log the API calls in the <API-M_HOME>/repository/logs/wso2-BotDetectedData.log file. The following is a sample log record.

JSON Schema Validation

Request Validation



New in APIM 3 series

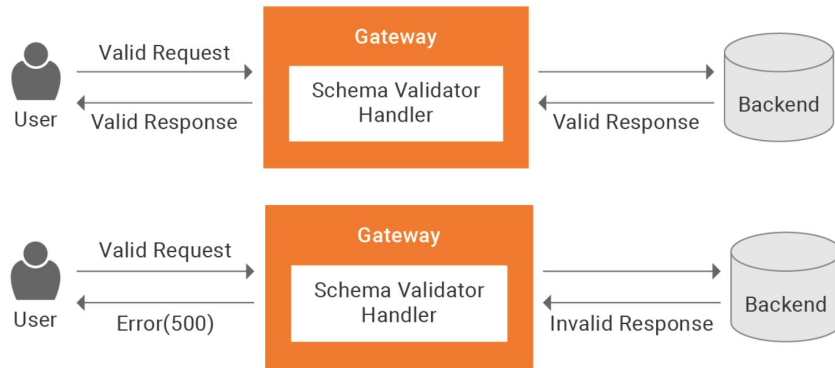
When creating APIs, it is possible to enable request/ response schema validation for API requests.

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

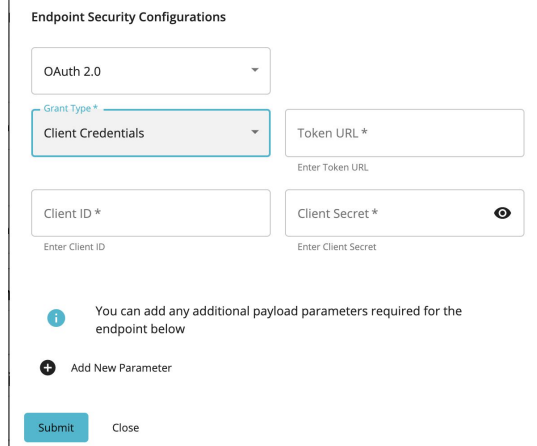
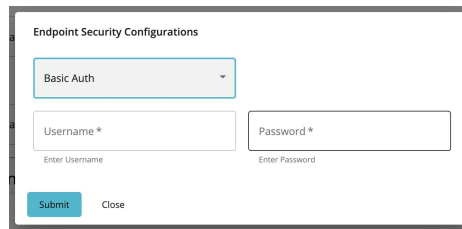
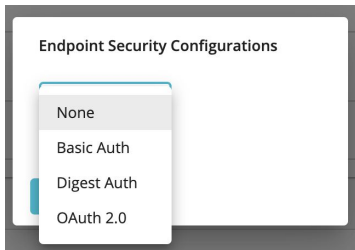
Response Validation



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



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.

Digest Authentication applies a hash function to the username and the password before sending them over the network.

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/3.2.0/install-and-setup/setup/security/logins-and-passwords/working-with-encrypted-passwords/#encrypting-secured-endpoint-passwords>

OAuth 2.0 - New in APIM 3.2.0

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 2 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, the loading time 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

i You do not have any certificates uploaded

Upload Certificate

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

Endpoint for the Certificate

Alias *
pizzashack

Alias for the Certificate

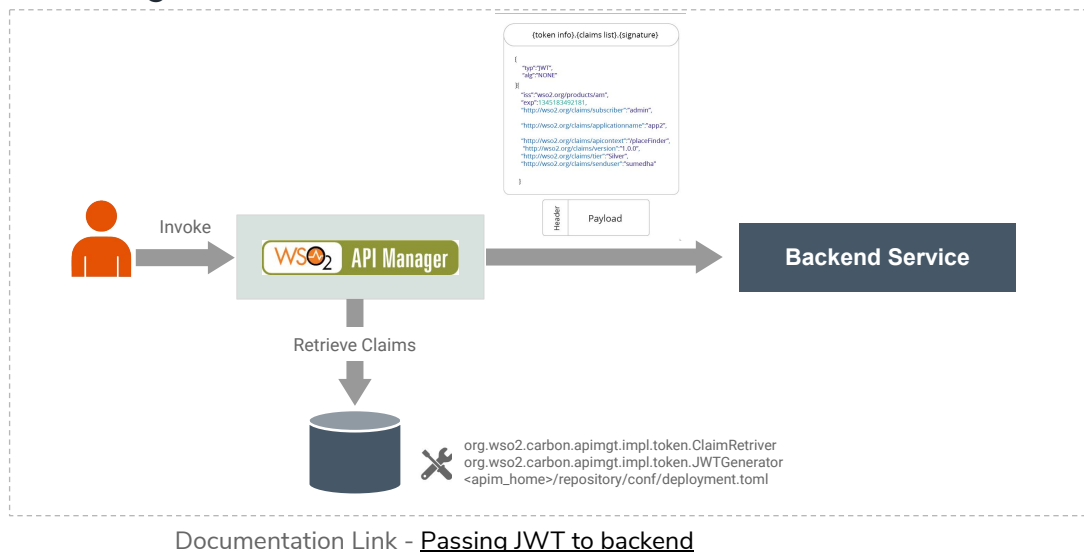
cert.crt

CLOSE SAVE

In the API Publisher portal,

- Click on an API and goto 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.



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

The screenshot displays the WSO2 API Manager interface. On the left, the 'API Definition' tab is active, showing the OpenAPI specification for 'PizzaShackAPI-1.0.0'. The specification includes details like title, version, contact, email, license, and a POST endpoint for creating a new order. On the right, the 'API Security Audit Report' is shown. It features an 'Audit Score and Summary' section with a score of 25 out of 100. Below this, there are sections for 'OpenAPI Format Requirements' (No Issues Found), 'Security' (listing 5 issues), and 'Data Validation' (listing 3 issues).

Severity	Description	Score Impact
CRITICAL	API includes HTTP responses in the client	-10
MEDIUM	The operation 'get' accepts access tokens sent as character	-3.33
MEDIUM	The operation 'post' accepts access tokens sent as character	-3.33
MEDIUM	The operation 'delete' accepts access tokens sent as character	-3.33
MEDIUM	The operation 'put' accepts access tokens sent as character	-3.33

Severity	Description	Score Impact
HIGH	Along with the name no maximum number of the items defined	-3.33
HIGH	Along with the name no maximum number of the items defined	-3.33
MEDIUM	Maximum number of items 'age' has no maximum defined	-3.33

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](#)