Quick Start Guide

WSO2 API Manager is a complete solution for designing and publishing APIs, creating and managing a developer community, and for securing and routing API traffic in a scalable manner. It leverages proven components from the WSO2 platform to secure, integrate and manage APIs. In addition, it integrates with the WSO2 analytics platform and provides out of the box reports and alerts, giving you instant insights into the APIs behavior.

Before you begin

Follow the instructions below to run WSO2 API-M.

Run on Docker

1. Run the command below to pull WSO2 API Manager 2.6.0 Docker image.

   ```bash
docker pull wso2/wso2am:2.6.0
   ```

2. Start a Docker container using the pulled image.

   ```bash
docker run -it -p 8280:8280 -p 8243:8243 -p 9443:9443 --name api-manager wso2/wso2am:2.6.0
   ```

   - Port 8280 and 8243 are used for API calls (HTTP and HTTPS respectively), and port 9443 is used for UI and internal services. For more information, see Docker Hub - wso2am.

Run the binary

1. Download and install JDK (AdoptOpenJDK 8, CorrettoJDK 8, OpenJDK 8, or Oracle JDK 1.8.*) and set the JAVA_HOME environment variable.

   For information on setting the JAVA_HOME environment variable for different operating systems, see Installing via the Installer.

2. Download the latest version of WSO2 API Manager.

3. Start WSO2 API Manager by going to the `<API-M_HOME>/bin` directory using the command-line and then executing `wso2server.bat` (for Windows) or `wso2server.sh` (for Linux).

Let's go through the use cases of the API Manager:

- Invoking your first API
- Understanding the API Manager concepts
- Deep diving into the API Manager
  - Creating users and roles
  - Creating an API from scratch
  - Adding API documentation
  - Adding interactive documentation
  - Versioning the API
  - Associating Scopes to API Resources
  - Publishing the API
  - Subscribing to the API
  - Invoking the API
  - Monitoring APIs and viewing statistics

Invoking your first API

Follow the steps in this section to quickly deploy a sample API, publish it, subscribe to it, and invoke it.

1. Open the API Publisher (https://<hostname>:9443/publisher) and sign in with admin/admin credentials.
2. Exit from API creation tutorial by clicking the close icon (X) on top right corner.

3. Click the **Deploy Sample API** button. It deploys a sample API called PizzaShackAPI into the API Manager.

   ![API Manager Screen](image)

   **This Deploy Sample API option is available only when there are no APIs in API Publisher. If you have already created an API, this option will not be available.**
4. Click PizzaShackAPI to open it.

5. Go to the Lifecycle tab and note that the State is PUBLISHED. The API is already published to the API Store.
6. Sign in to the API Store (https://<hostname>:9443/store) with the admin/admin credentials and click on the PizzaShackAPI API.

7. Select the default application and an available tier, and click Subscribe.
8. When the subscription is successful, click **View Subscriptions** on the information message that appears. Click the **Production Keys** tab and click **Generate Keys** to generate an access token to invoke the API.

You have now successfully subscribed to an API. Let's invoke the API using the integrated Swagger-based API Console.

9. Click the **APIs** menu again and click the **PizzaShackAPI** to open it. When the API opens, click its **API Console** tab.

Expand the GET method (which retrieves the menu) and click **Try it out**.
Understanding the API Manager concepts

Before we look into the API management activities in detail, let's take a look at the basic API management concepts.

Components

- **API Publisher**: Enables API providers to publish APIs, share documentation, provision API keys and gather feedback on features, quality and usage. You access the Web interface via https://<Server Host>:9443/publisher.
- **API Store (Developer Portal)**: Enables API consumers to self register, discover and subscribe to APIs, evaluate them and interact with API publishers. You access the Web interface via https://<Server Host>:9443/store.
- **API Gateway**: Secures, protects, manages, and scales API calls. It is a simple API proxy that intercepts API requests and applies policies such as throttling and security checks. It is also instrumental in gathering API usage statistics. The Web interface can be accessed via https://<Server Host>:9443/carbon.
- **Key Manager**: Handles all security and key-related operations. The API Gateway connects with the Key Manager to check the validity of subscriptions, OAuth tokens, and API invocations. The Key Manager also provides a token API to generate OAuth tokens that can be accessed via the Gateway.
- **Traffic Manager**: Helps users to regulate API traffic, make APIs and applications available to consumers at different service levels and secures APIs against security attacks. The Traffic Manager features a dynamic throttling engine to process throttling policies in real-time.
- **WSO2 API Manager Analytics**: Provides a host of statistical graphs and an alerting mechanism on predetermined events.
Users and roles

The API manager offers three distinct community roles that are applicable to most enterprises:

- **Creator**: A creator is a person in a technical role who understands the technical aspects of the API (interfaces, documentation, versions, how it is exposed by the Gateway, etc.) and uses the API publisher to provision APIs into the API Store. The creator uses the API Store to consult ratings and feedback provided by API users. Creators can add APIs to the store but cannot manage their life cycle (e.g., make them visible to the outside world.)
- **Publisher**: A publisher manages a set of APIs across the enterprise or business unit and controls the API life cycle and monetization aspects.
- **Consumer**: A consumer uses the API Store to discover APIs, see the documentation and forums, and rate/comment on the APIs. Consumers subscribe to APIs to obtain API keys.

API lifecycle

An API is the published interface, while the service is the implementation running in the backend. APIs have their own lifecycles that are independent of the backend services they rely on. This lifecycle is exposed in the API Publisher and is managed by the publisher role.

The following stages are available in the default API life cycle:

- **CREATED**: API metadata is added to the API Store, but it is not visible to subscribers yet, nor deployed to the API Gateway.
- **PROTOTYPED**: The API is deployed and published in the API Store as a prototype. A prototyped API is usually a mock implementation made public in order to get feedback about its usability. Users can try out a prototyped API without subscribing to it.
- **PUBLISHED**: The API is visible in the API Store and available for subscription.
- **DEPRECATED**: The API is still deployed in the API Gateway (i.e., available at runtime to existing users) but not visible to subscribers. You can deprecate an API automatically when a new version of it is published.
- **RETIRED**: The API is unpublished from the API Gateway and deleted from the Store.
- **BLOCKED**: Access to the API is temporarily blocked. Runtime calls are blocked, and the API is not shown in the API Store anymore.
Applications

An application is primarily used to decouple the consumer from the APIs. It allows you to do the following:

- Generate and use a single key for multiple APIs.
- Subscribe multiple times to a single API with different SLA levels.

You create an application to subscribe to an API. The API Manager comes with a default application, and you can also create as many applications as you like.

Throttling tiers

Throttling tiers are associated with an API at subscription time and can be defined at an API-level, resource-level, subscription-level and application-level (per token). They define the throttling limits enforced by the API Gateway, e.g., 10 TPS (transactions per second). The final throttle limit granted to a given user on a given API is ultimately defined by the consolidated output of all throttling tiers together. The API Manager comes with three predefined tiers for each level and a special tier called Unlimited, which you can disable by editing the `<ThrottlingConfigurations>` element of the `<API-M_HOME>/repository/conf/api-manager.xml` file.

In API Manager 2.0.0 onwards, Advanced Throttling is enabled by default with following configuration in `<API-M_HOME>/repository/conf/api-manager.xml`.

```
<ThrottlingConfigurations>
    <EnableAdvanceThrottling>true</EnableAdvanceThrottling>
.......
</ThrottlingConfigurations>
```

If you are disabling Advanced Throttling in any case by setting the value of `<EnableAdvanceThrottling>` false, Advanced Throttling is disabled and basic Throttling mechanism is enabled thereafter. In such a scenario, if you want to disable the Unlimited Throttling tier of basic Throttling configurations, you need to disable it under `<TierManagement>` by setting `<EnableUnlimitedTier>` to false.

```
<TierManagement>
    <EnableUnlimitedTier>true</EnableUnlimitedTier>
</TierManagement>
```

Predefined Subscription Tiers.

<table>
<thead>
<tr>
<th>Throttling Tier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unlimited</td>
<td>Allows unlimited requests</td>
</tr>
<tr>
<td>Gold</td>
<td>Allows 5000 requests per minute</td>
</tr>
<tr>
<td>Silver</td>
<td>Allows 2000 requests per minute</td>
</tr>
<tr>
<td>Bronze</td>
<td>Allows 1000 requests per minute</td>
</tr>
</tbody>
</table>

API keys

The API Manager supports two scenarios for authentication:

- An access token is used to identify and authenticate a whole application.
- An access token is used to identify the final user of an application (for example, the final user of a mobile application that is deployed on many devices).

Application access token: Application access tokens are generated by the API consumer and must be passed in the incoming API requests. The API Manager uses the OAuth2 standard to provide key management. An API key is a simple string that you pass with an HTTP header (e.g., "Authorization: Bearer NtBqkXEOKlu0H1alfQODw06IX4a,"), and it works equally well for SOAP and REST calls.

Application access tokens are generated at the application level and valid for all APIs that you associate to the application. These tokens have a fixed expiration time, which is set to 60 minutes by default. You can change this to a longer time, even for several weeks. Consumers can regenerate the access token directly from the API Store. To change the default expiration time which is 60 minutes by default, you open the `<API-M_HOME>/repository/conf/identity/identity.xml` file and change the value of the element `<AccessTokenDefaultValidityPeriod>`. If you set a negative value, the token never expires. Changes to this value are applied only to the new applications that you create.
**Application user access token:** You generate access tokens on demand using the Token API. In case a token expires, you use the Token API to refresh it.

The Token API takes the following parameters to generate the access token:

- **Grant Type**
- **Username**
- **Password**
- **Scope**

To generate a new access token, you issue a Token API call with the above parameters where `grant_type=password`. The Token API then returns two tokens: an access token and a refresh token. The access token is saved in a session on the client side (the application itself does not need to manage users and passwords). On the API Gateway side, the access token is validated for each API call. When the token expires, you refresh the token by issuing a token API call with the above parameters where `grant_type=refresh_token` and passing the refresh token as a parameter.

**API resources**

An API is made up of one or more resources. Each resource handles a particular type of request and is analogous to a method (function) in a larger API. API resources accept the following optional attributes:

- **verbs:** Specifies the HTTP verbs a particular resource accepts. Allowed values are GET, POST, PUT, DELETE, PATCH, HEAD, and OPTIONS. You can give multiple values at once.
- **url-mapping:** A URL mapping defined as per the servlet specification (extension mappings, path mappings, and exact mappings).
- **Throttling tiers:** Limits the number of hits to a resource during a given period of time.
- **Auth-Type:** Specifies the Resource level authentication along the HTTP verbs. Auth-type can be None, Application, Application User, or Application & Application User.
  - None: Can access the particular API resource without any access tokens.
  - Application: An application access token is required to access the API resource.
  - Application User: A user access token is required to access the API resource.
  - Application & Application User: An application access token together with a user access token is required to access the API resource.

---

**Deep diving into the API Manager**

Let's take a look at the typical API management activities in detail:

- Creating users and roles
- Creating an API from scratch
- Adding API documentation
- Adding interactive documentation
- Versioning the API
- Associating Scopes to API Resources
- Publishing the API
- Subscribing to the API
- Invoking the API
- Monitoring APIs and viewing statistics

**Creating users and roles**

In users and roles, we introduced a set of users who are commonly found in many enterprises. Let’s see how you can sign in to the Management Console as an admin and create these roles.

2. Click **Add** in the **Users and Roles** section under the **Main** menu.

![Dashboard](image)

The creator, publisher, and subscriber roles are available by default as shown below.
3. Click **Add New Role**.

![Add New Role](image)

4. Give the role name as `creator` and click **Next**.

![Add New Role](image)

5. A list of permissions opens. Select the following and click **Finish**.
   - All Permissions > Admin Permissions > Configure > Governance and all underlying permissions
   - All Permissions > Admin Permissions > Login
   - All Permissions > Admin Permissions > Manage > API > Create
   - All Permissions > Admin Permissions > Manage > Resources > Govern and all underlying permissions
5. Similarly, create the publisher role with the following permissions.

- All Permissions > Admin Permissions > Login
- All Permissions > Admin Permissions > Manage > API > Publish

7. Note that the API Manager comes with the subscriber role available by default. It has the following permissions:

- All Permissions > Admin Permissions > Login
- All Permissions > Admin Permissions > Manage > API > Subscribe
8. The roles you added (creator and publisher) are now displayed under Roles.

For more details Add roles and permission assign to roles, see Adding User Roles. Let's create users for each of the roles.

9. Click Add in the Users and Roles section under the Main menu.

10. Click Add New User.

11. Give the username/password and click Next. For example, let's create a new user by the name apipublisher.
12. Select the role you want to assign to the user (e.g., publisher) and click Finish.

![Add User](image)

13. Similarly, create a new user by the name apicreator and assign the creator role.

**Creating an API from scratch**

Let's create an API from scratch.

1. Sign in to the API Publisher (https://<hostname>:9443/publisher) as apicreator.
2. In the APIS menu, click Add New API.

![API Publisher](image)

3. Select the option to design a new API and click Start Creating. Let's get started!

4. Give the information in the table below.

<table>
<thead>
<tr>
<th>Field</th>
<th>Sample value</th>
</tr>
</thead>
</table>
Click Add and then click Next: Implement > to move on to the next page.

5. Select the Managed API option.
   PhoneVerification: /phoneverify/1.0.0

For instructions on how to Implement Prototyped API, see Create a Prototyped API with an Inline Script.

Give the following information and click Next:Manage > once you are done.

<table>
<thead>
<tr>
<th>Field</th>
<th>Sample value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Endpoint type</td>
<td>HTTP/REST Endpoint</td>
</tr>
<tr>
<td>Production endpoint</td>
<td>Endpoint is <a href="http://ws.cdyne.com/phoneverify/phoneverify.asmx">http://ws.cdyne.com/phoneverify/phoneverify.asmx</a>. To verify the URL, click the Test button next to it. In this example, we use a phone validation service exposed by the Cdyne services provider. This service has SOAP and REST interfaces. This sample service has two operations: CheckPhoneNumber and CheckPhoneNumbers.</td>
</tr>
<tr>
<td>Sandbox endpoint</td>
<td>Endpoint is <a href="http://ws.cdyne.com/phoneverify/phoneverify.asmx">http://ws.cdyne.com/phoneverify/phoneverify.asmx</a>. To verify the URL, click the Test button next to it.</td>
</tr>
</tbody>
</table>
6. Provide the following information in the Manage tab. Leave default values for the rest of the parameters in the UI.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subscription Tiers</td>
<td>&lt;Select all available tiers&gt;</td>
<td>The API can be available for subscription at different levels of service. They allow you to limit the number of successful hits to an API during a given period of time.</td>
</tr>
</tbody>
</table>
Once you are done, click Save.

Adding API documentation

1. In the APIS menu, click the thumbnail of the API to open it.
2. Click on the API’s Docs tab and click Add New Document.

3. The document options appear. Note that you can create documentation inline, via a URL, or as a file. For inline documentation, you can edit the content directly from the API publisher interface. You get several documents types:

- How To
- Samples and SDK
- Public forum / Support forum (external link only)
- API message formats
- Other
4. Create a 'How To' named PhoneVerification, specifying in-line content as the source and optionally entering a summary. When you have finished, click Add Document.

PhoneVerification - 1.0.0

![Add Document dialog]

5. Once the document is added, click Edit Content to open an embedded editor.

PhoneVerification - 1.0.0

![Documents list with 'Edit Content' button]
6. Enter your API's documentation and click Save and Close.

Adding interactive documentation

WSO2 API Manager has an integrated Swagger UI, which is part of the Swagger project.

Swagger is a 100% open source, standard, language-agnostic specification and a complete framework for describing, producing, consuming, and visualizing RESTful APIs, without the need of a proxy or third-party services. Swagger allows consumers to understand the capabilities of a remote service without accessing its source code and interact with the service with a minimal amount of implementation logic. Swagger helps describe services in the same way that interfaces describe lower-level programming code.

The Swagger UI is a dependency-free collection of HTML, JavaScript, and CSS that dynamically generates documentation from a Swagger-compliant API. Swagger-compliant APIs give you interactive documentation and more discoverability. The Swagger UI has YAML code, and its UI facilitates easier code indentation, provides keyword highlighting, and shows syntax errors on the fly. You can add resource parameters, summaries and descriptions to your APIs using the Swagger UI and it has the facility to download your API definition as YAML or JSON file. Go to the Swagger 2.0 specification for more information.

1. Open the API Publisher (https://<hostname>:9443/publisher) and sign in as apicreator.
2. Click the Edit icon for the PhoneVerification API. This opens the API in its edit mode.

3. Click the Edit Source button under the API Definition section.

4. The API definition as a YAML code opens in a separate page. Expand its GET method, add the following parameters and click Apply Changes.

```yaml
parameters:
  - in: query
    name: PhoneNumber
    description: Give the phone number to be validated
    type: string
    required: true
  - in: query
    name: LicenseKey
    description: Give the license key as 0 for testing purpose
    type: string
    required: true
```
```json
swagger: '2.0'
paths:
  /CheckPhoneNumber:
    get:
      responses:
        '200':
          description: ''
          x-auth-type: Application & Application User
          x-throttling-tier: Unlimited
          parameters:
            - in: query
              name: PhoneNumber
              description: Give the phone number to be validated
              type: string
              required: true
            - in: query
              name: LicenseKey
              description: Give the license key as 0 for testing purpose
              type: string
              required: true
    post:
      responses:
        '200':
          description: ''
          parameters:
            - name: Payload
              description: Request Body
              required: false
              in: body
              schema:
                type: object
                properties:
                  payload:
                    type: string
          x-auth-type: Application & Application User
          x-throttling-tier: Unlimited
info:
  title: PhoneVerification
  version: 1.0.0
```
5. Back in the API Publisher, note that the changes you did appear in the API Console's UI. You can add more parameters and edit the summary/descriptions using the API Publisher UI as well. Once done, click **Save**.

Versioning the API

Let's create a new version of this API.

1. Sign in to the API Publisher as **apicreator** if you are not logged in already.
2. Click the **PhoneVerification** API to open it and then click **Create New Version.**
3. Give a new version number (e.g., 2.0.0) and click **Done**.

**PhoneVerification - 1.0.0**

Create New Version

- **New Version**: 2.0.0
- **Make this the Default Version**: False

*Info!*
No default version defined for the current API

**Done**  **Cancel**

4. Note that the new version of the API is created in the API Publisher.

**Associating Scopes to API Resources**

Different API resources can be associated with different user roles. For an example, consider the following resources and the operations:

<table>
<thead>
<tr>
<th>Method</th>
<th>URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/orders</td>
</tr>
<tr>
<td>POST</td>
<td>/orders</td>
</tr>
<tr>
<td>GET</td>
<td>/items</td>
</tr>
<tr>
<td>POST</td>
<td>/items</td>
</tr>
</tbody>
</table>

In order to map a scope to an API resource, the following should be done:

1. API creator should first create scopes, by clicking on Add Scopes in the Manage tab.

2. Fill the scope related information in the dialog that pops up. Note that Scope Key and Roles are the most important attributes. Click on Add Scope button on the right hand side bottom.
3. Now the scope with key 'item_view' is added with roles manager and agent. To associate this scope with the get operation on the /time resource, click on the +Scope sign on the right hand side of the resource.

4. From the drop down menu that appears, select the scope name and click on the tick sign to the right. Now the scope will be associated with the GET operation on the resource /item.

See Scope Management with OAuth Scopes for an in depth example.

**Publishing the API**

1. Sign in to the API Publisher as the apipublisher user that you created earlier in this guide, and click the PhoneVerification API's version 2.0.0.

   ![PhoneVerification API Version 2.0.0](image)

   The API opens. Go to its Lifecycle tab and click Publish.

   ![Lifecycle Tab](image)

   The check boxes mean the following:
   - **Require re-subscription when publish the API**: Invalidates current user subscriptions, forcing users to subscribe again.
   - **Deprecate old versions after publish the API**: If selected, any prior versions of the API that are published will be set to the DEPRECATED state automatically.
3. Go to the API Store (https://<hostname>:9443/store) using your browser and note that the PhoneVerification 2.0.0 API is visible under the APIs menu.

Subscribing to the API

1. Go to the API Store (https://<hostname>:9443/store) and create an account using the Sign-up link.

   Users who sign-up through the API Store are assigned the subscriber role by default. Therefore, you do not need to specify the role through the management console to be able to subscribe to an API.

2. Fill the details in the Sign Up form appears and click Sign Up.

   Users who registered with the API Store Signup can be view by login to the Management Console (https://localhost:9443/carbon) and accessing Users and Roles > Users > List.
Details entered in the sign up will be updated in the default profile related to each user in the management console.

3. After signing up, sign in to the API Store and click the PhoneVerification 2.0.0 API that you published earlier.
4. Note that you can now see the subscription options. Select the default application and the Bronze tier. Click Subscribe.

5. Once the subscription is successful, click View Subscriptions in the information message that appears to review your subscriptions.

Subscription Successful

You have successfully subscribed to the API.
6. Click the Production Keys tab of the application and then click Generate Keys to generate an access token that you use later to invoke the API. If you have already generated keys before, click Re-generate.

Tip: You can set a token validity period in the given text box. By default, it is set to one hour. If you set a minus value (e.g., -1), the token will never expire.

You are now successfully subscribed to an API. Let's invoke it.

Invoking the API

1. Click the APIs menu in the API Store and then click on the API that you want to invoke. When the API opens, go to its API Console tab.
2. Expand the GET method of the resource `CheckPhoneNumber`. Note the parameters that you added when creating the interactive documentation now appear with their descriptions so that as a subscriber, you know how to invoke this API.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
<th>Parameter Type</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PhoneNumber</td>
<td>[required]</td>
<td>Give the phone number to be validated</td>
<td>query</td>
<td>string</td>
</tr>
<tr>
<td>LicenseKey</td>
<td>[required]</td>
<td>Give the license key as 0 for testing purpose</td>
<td>query</td>
<td>string</td>
</tr>
</tbody>
</table>

3. Give sample values for the `PhoneNumber` and `LicenseKey` and click **Try it out** to invoke the API.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
<th>Parameter Type</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PhoneNumber</td>
<td>18000785402</td>
<td>Give the phone number to be validated</td>
<td>query</td>
<td>string</td>
</tr>
<tr>
<td>LicenseKey</td>
<td>0</td>
<td>Give the license key as 0 for testing purpose</td>
<td>query</td>
<td>string</td>
</tr>
</tbody>
</table>

4. Note the response for the API invocation. Since we used a valid phone number in this example, the response is valid.

**Response Body**

```xml
  <soap:Body>
    <ns1:checkPhoneNumberResponse xmlns:ns1="http://example.com">
      <checkPhoneNumberResult>true</checkPhoneNumberResult>
    </ns1:checkPhoneNumberResponse>
  </soap:Body>
</soap:Envelope>
```

**Response Code**: 200

**Response Headers**

```plaintext
{
  "cache-control": "no-cache",
  "content-type": "text/xml; charset=utf-8",
  "x-generator": "WSO2 APIM",
  "x-apikey": "-"
}
```

You have invoked an API using the API Console.

**Troubleshooting**

When using the API Console, the web browser sends an HTTPS request to the Gateway. If the certificate in the Gateway is not CA signed, the browser will not accept it. Therefore, you may get the following error.

```
ERROR - SourceHandler I/O error: Received fatal alert: unknown_ca
javax.net.ssl.SSLException: Received fatal alert: unknown_ca
```

As a workaround, you can access the Gateway URL on a new browser tab and trust the certificate from the browser.
## Monitoring APIs and viewing statistics

Both the API publisher and store provide several statistical dashboards.

### API Publisher statistics

#### Statistics

<table>
<thead>
<tr>
<th>APIs</th>
<th>Applications</th>
<th>Subscriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Published APIs Over Time</td>
<td>App Throttled Requests</td>
<td>API Subscriptions</td>
</tr>
<tr>
<td>API Usage</td>
<td>Applications Created Over Time</td>
<td>Developer Signups Over Time</td>
</tr>
<tr>
<td>API Response Times</td>
<td></td>
<td>Subscriptions Created Over Time</td>
</tr>
<tr>
<td>API Last Access Times</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage by Resource Path</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage by Destination</td>
<td></td>
<td></td>
</tr>
<tr>
<td>API Usage Comparison</td>
<td></td>
<td></td>
</tr>
<tr>
<td>API Throttled Requests</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Faulty Invocations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>API Latency Time</td>
<td></td>
<td></td>
</tr>
<tr>
<td>API Usage Across Geo Locations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>API Usage Across User Agent</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### API Store statistics

![WSO2 API Store Analytics](image)
The steps below explain how to configure WSO2 API Manager Analytics with the API Manager. The statistics in these dashboards are based on data from WSO2 APIM Analytics which is similar to WSO2 Stream Processor (WSO2 SP).

Let's do the configurations first.

1. To enable Analytics, open the `<API-M_HOME>/repository/conf/api-manager.xml` file and set the `Enabled` property under `<Analytics>` to `true` as shown below. Save this change.

   `<Enabled>true</Enabled>`

2. Start the WSO2 APIM Analytics server, and then start the WSO2 API Manager server.

   To avoid connection errors during API Manager startup, start WSO2 APIM Analytics before WSO2 API Manager.

   a. On Windows: `worker.bat --run`
   b. On Linux/Mac OS: `sh worker.sh`

   By default, WSO2 API Manager has a port offset of 0 (no port offset) and WSO2 API Manager Analytics has an offset of 1.

3. Invoke several APIs to generate some statistical data and wait a few seconds.
4. Connect to the API Publisher as a creator and click one of the statistical dashboards available in the Statistics menu. For example,

   **Overall API Subscriptions (Across All Versions)**

   ![Overall API Subscriptions](image)

   The Statistics menu is available for API creators and shows statistics of all APIs. Additionally, API creators can also see the following:
   - Statistics of the APIs created by them by selecting the My APIs option in the drop down menu above each table or graph.
   - The subscriptions of each API by clicking Manage Subscriptions.
   - The alerts that can be configured for their APIs by clicking Manage Alert Types.

This concludes the API Manager quick start. You have set up the API Manager and gone through the basic use cases of the product. For more advanced use cases, see the Tutorials, Using WSO2 API Manager and Admin Guide of the API Manager documentation.