***WSO2 API MANAGER***

**Table Of Contents**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Contents** | **Page No** |
| 1 | Getting Started with Wso2 | 1 |
| 2 | Wso2 API Manager | 2 |
| 3 | Wso2 APIM Analytics | 15 |
| 4 | Configure APIM Analytics with API Manager | 15 |
| 5 | Statistics Report | 17 |
| 6 | WSo2 Identity server | 24 |
| 7 | Configure Wso2 Identity server with Wso2 API Manager | 24 |
| 8 | Getting Started with Throtling | 32 |

1. Getting Started with Wso2

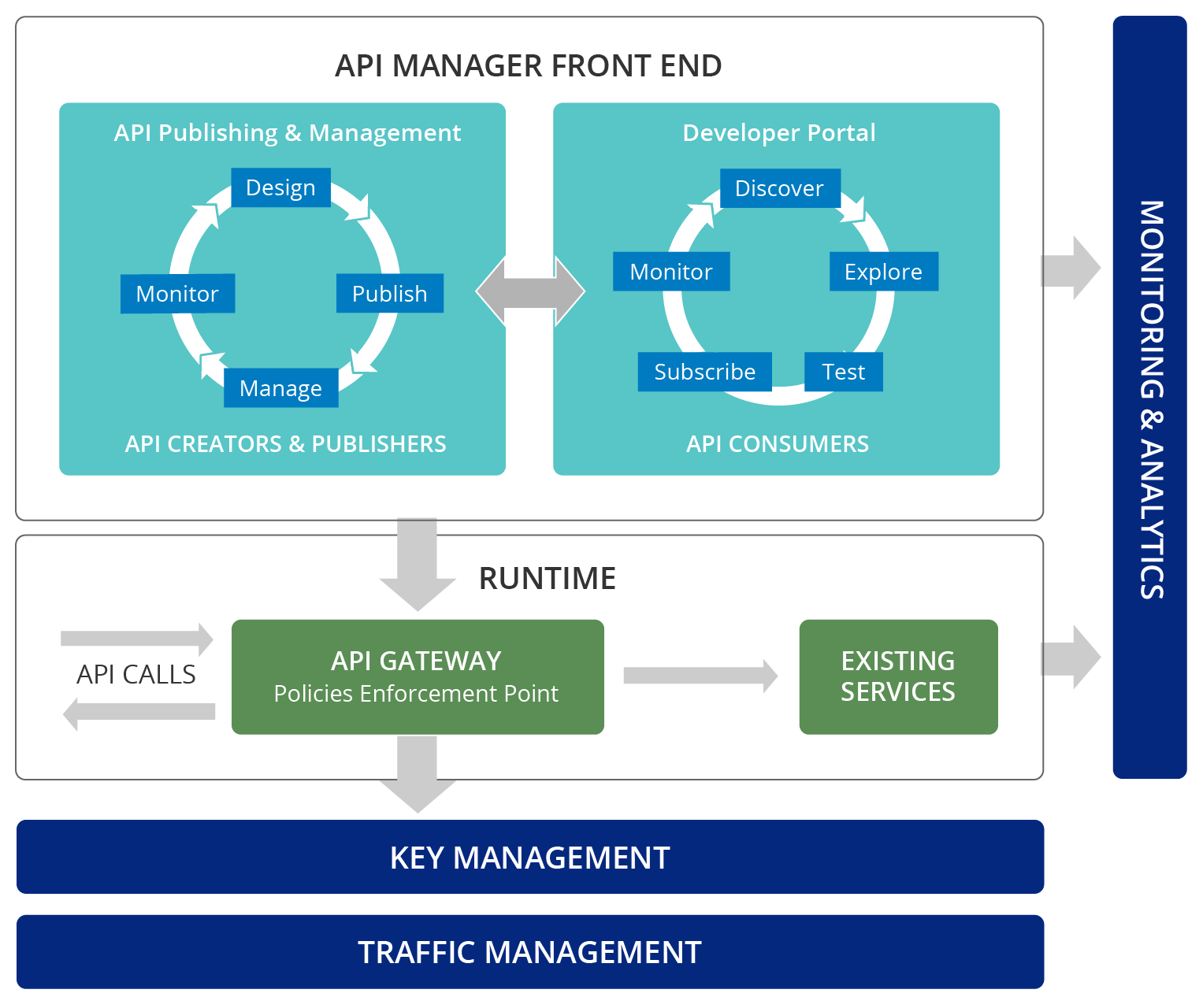
**WSO2** (sometimes stylized as WSO2) is an open source technology company providing service-oriented architecture (SOA) middleware. It is best known for its enterprise service bus, application programming interface management, governance and analytics offerings, some of which are used by eBay, Boeing, Experian.

It offers wide range of products for each serves individual purpose ,Some of them are as follows:

1. WSo2 API Manager.
2. Wso2 Analytics Server.
3. WSo2 Identity Server.

*2.Wso2 API Manager*

WSO2 API Manager is a complete solution for designing and publishing APIs, creating and managing a developer community, and for scalable routing API traffic. It leverages proven, production-ready integration, security, and governance components from the WSO2 Enterprise Service Bus, WSO2 Identity Server, and WSO2 Governance Registry. In addition, it leverages the WSO2 Data Analytics Server for analytics, giving you instant insight into APIs behavior.



*Fig : API Manager Architecture.*

***API Manager components .***

1. *API Publisher*
2. *API Subscriber*
3. *API GateWay*
4. *Key Manager*

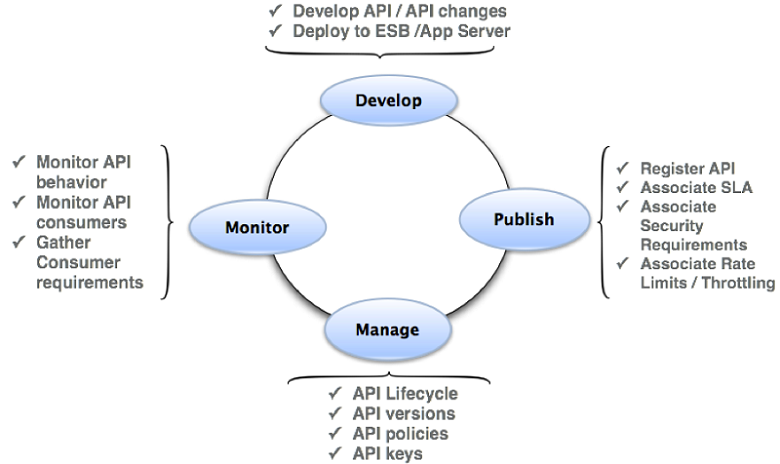
**API Publisher**

API development is usually done by someone who understands the technical aspects of the API, interfaces, documentation, versions etc., while API management is typically carried out by someone who understands the business aspects of the APIs. In most business environments, API development is a responsibility that is distinct from API publication and management.

WSO2 API Manager provides a simple Web interface called **WSO2 API Publisher** for API development and management. It is a structured GUI designed for API creators to develop, document, scale and version APIs, while also facilitating more API management-related tasks such as publishing API, monetization, analyzing statistics, and promoting.

The API Publisher URL is **https://<YourHostName>:9443/publisher** and it is accessible on HTTPS only. The default credentials are admin/admin.

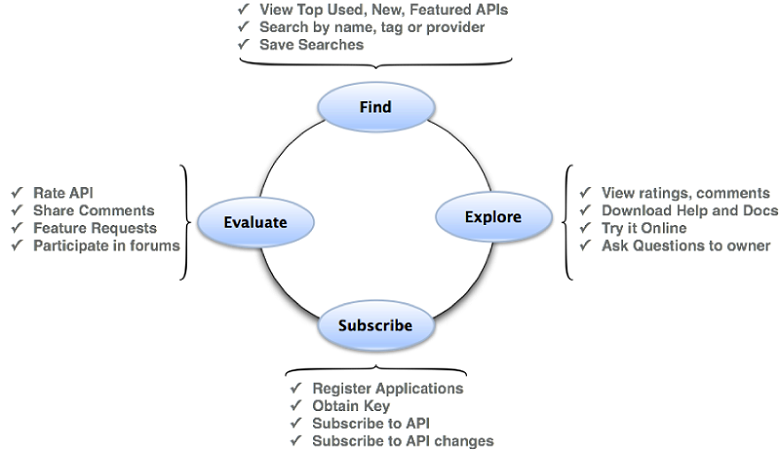
The diagram below shows the common lifecycle activities of an API developer/manager:

****

**API Store (Developer Portal)**

The API Store Web application provides a collaborative interface for API publishers to host and advertise their APIs and for API consumers to [self-register](https://docs.wso2.com/display/AM200/Customizing+the+API+Store),discover, evaluate, subscribe to and use secured, protected, authenticated APIs. The API Store URL is **https://<YourHostName>:9443/store** and it is accessible on HTTPS only. The default credentials are admin/admin.

The diagram below shows common API consumer lifecycle activities:



**API Gateway**

A runtime, backend component (an API proxy) developed using WSO2 ESB. API Gateway secures, protects, manages, and scales API calls. It intercepts API requests, applies policies such as throttling and security using handlers and manages API statistics. Upon validation of a policy, the Gateway passes Web service calls to the actual backend. If the service call is a token request, the Gateway passes it directly to the [Key Manager](https://docs.wso2.com/display/AM200/Key+Concepts#KeyConcepts-KeyManager).

When the API Manager is running, you can access the Gateway using the URL <https://localhost:9443/carbon>.

You integrate a monitoring and analytics component to the API Manager by [configuring WSO2 API Manager Analytics](https://docs.wso2.com/display/AM200/Configuring+APIM+Analytics). This component provides reports, statistics and graphs on the APIs deployed in WSO2 API Manager. You can then configure alerts to monitor these APIs and detect unusual activity, manage locations via geo location statistics and carry out detailed analysis of the logs.

**Key Manager**

Manages all clients, security and access token-related operations. The Gateway connects with the Key Manager to check the validity of OAuth tokens, subscriptions and API invocations. When a subscriber creates an application and generates an access token to the application using the API Store, the Store makes a call to the API Gateway, which in turn connects with the Key Manager to create an OAuth client and obtain an access token. Similarly, to validate a token, the API Gateway calls the Key Manager, which fetches and validates the token details from the database.

**Users and roles**

The API Manager offers four distinct community roles that are applicable to most enterprises:

* **Admin**: The API management provider who hosts and manages the  API Gateway . S/he is responsible for creating user roles in the system, assign them roles, managing databases, security etc. The Admin role is available by default with credentials admin/admin.
* **Creator**: a creator is a person in a technical role who understands the technical aspects of the API (interfaces, documentation, versions 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. Creator can add APIs to the store but cannot manage their lifecycle.
* **Publisher** : a publisher manages a set of APIs across the enterprise or business unit and controls the API lifecycle, subscriptions and monetization aspects. The publisher is also interested in usage patterns for APIs and has access to all API statistics.
* **Subscriber** : a subscriber uses the API store to discover APIs, read the documentation and forums, rate/comment on the APIs, subscribes to APIs, obtain access tokens and invoke the APIs.

**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 to the backend services they rely on. This lifecycle is exposed in the API publisher Web interface and is managed by the API publisher role.

The following stages are available in the default API lifecycle:

* **CREATED:** API metadata is added to the API Store, but it is not deployed in the API gateway and therefore, is not visible to subscribers in the API Store.
* **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 invoke the API without a subscription.
* **PUBLISHED**: The API is visible in the API Store and available for subscription.
* **DEPRECATED:** When an API is deprecated, new subscriptions are disabled. But the API is still deployed in the Gateway and is available at runtime to existing subscribers. Existing subscribers can continue to use it as usual until the API is retired.
* **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 a logical collection of APIs. An application is primarily used to decouple the consumer from the APIs. It allows you to :

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

You subscribe to APIs through an application. Applications are available at different SLA levels, and have application-level throttling tiers engaged in them. A throttling tier determines the maximum number of calls you can make to an API during a given period of time.

The API Manager comes with a pre-created default application, which allows unlimited access by default. You can also [create](https://docs.wso2.com/display/AM200/Subscribe+to+an+API) your own applications.

**Access tokens**

An**access token**is a simple string that is passed as an HTTP header of a request. For example, "Authorization: Bearer NtBQkXoKElu0H1a1fQ0DWfo6IX4a." Access tokens authenticate API users and applications, and ensure better security (e.g., prevent**DoS attacks**). If a token that is passed with a request is invalid, the request is discarded in the first stage of processing. Access tokens work equally well for SOAP and REST calls.

***Installation Steps***

1. Install [Oracle Java SE Development Kit (JDK)](http://java.sun.com/javase/downloads/index.jsp) version 1.7.\* or 1.8.\* and set the JAVA\_HOME environment variable.
2. [Download](http://wso2.com/api-management/try-it) WSO2 API Manager.
3. Start the API Manager by going to <APIM\_HOME>/bin using the command-line and executing wso2server.bat (for Windows) or wso2server.sh (for Linux.)

***Create and Publish API***

**API creation** is the process of linking an existing backend API implementation to the API Publisher so that you can manage and monitor the API's lifecycle, documentation, security, community and subscriptions. Alternatively, you can provide the API implementation in-line in the API Publisher itself.

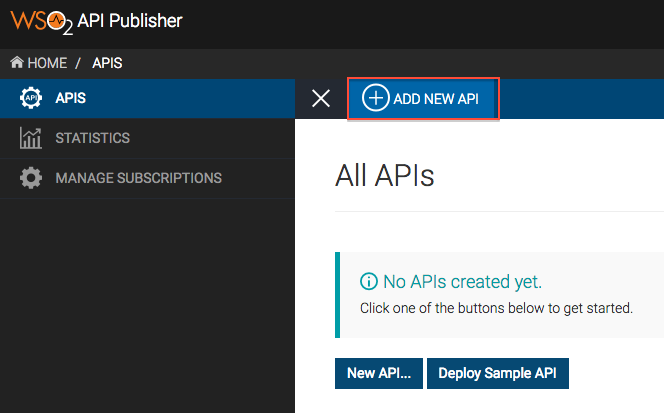
The steps below shows you how to create a new API.

1. Sign in to the WSO2 API Publisher.  
   https://<hostname>:9443/publisher

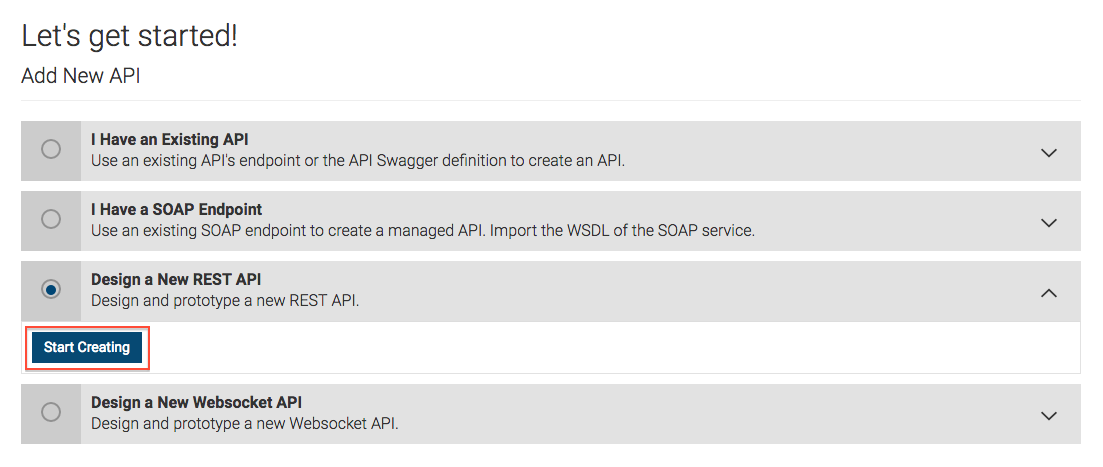
We have implemented a shopping Cart Application with Micro Service Architecture. So we have some 5 micro services communicating with each Other .

Let’s see One API how we can configure in wso2.

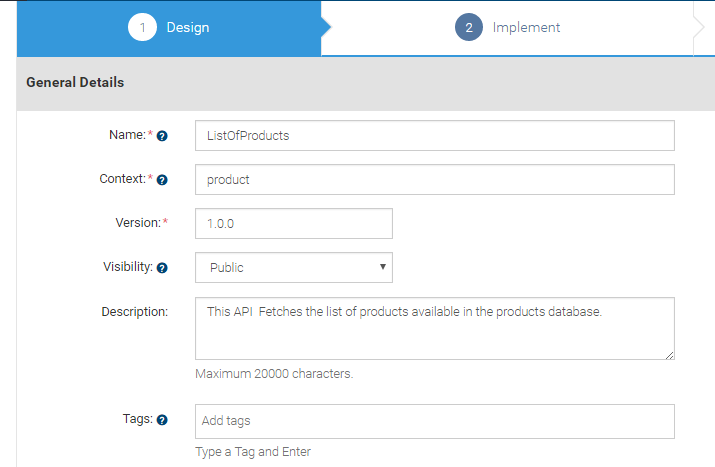
2.In the **APIS** menu, click **Add New API**



3.Select **Design New REST API** and click **Start Creating.**

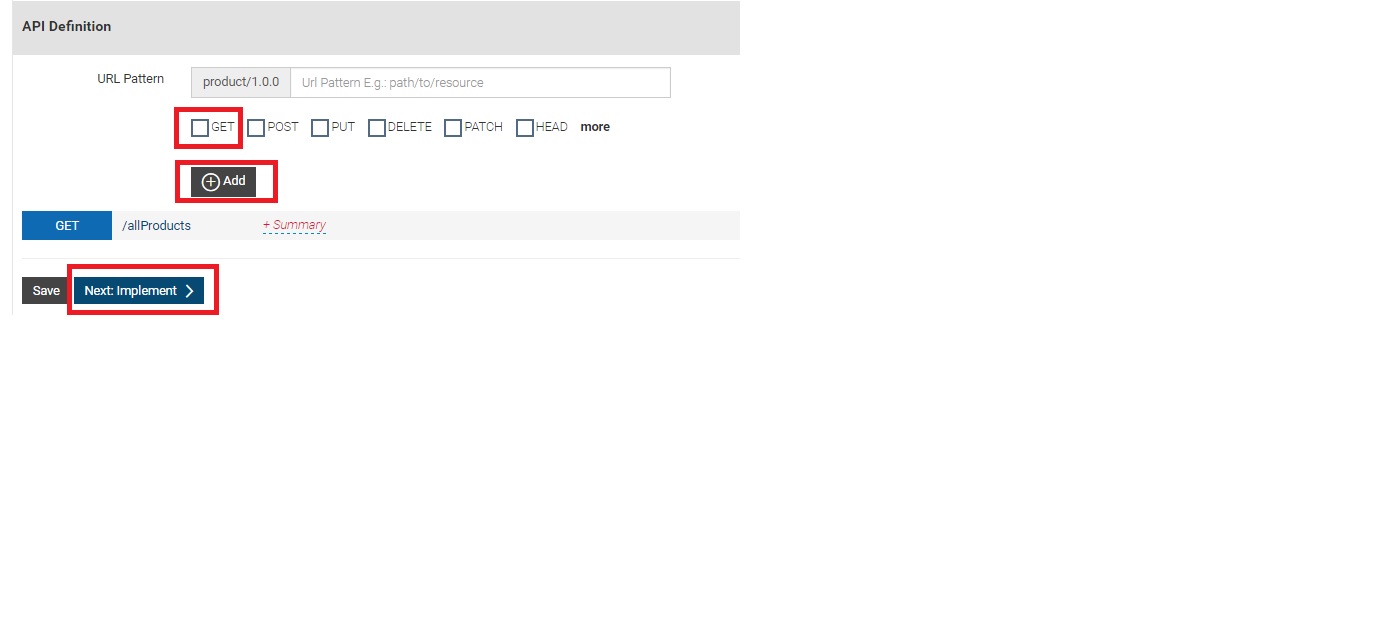


1. Give the below information in the General Details Section.



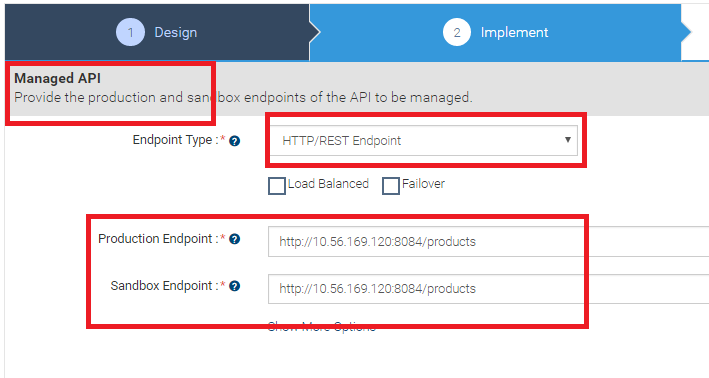
5.After you add the resource, click it's GET method to expand it and

Once done, click **Next: Implement >**



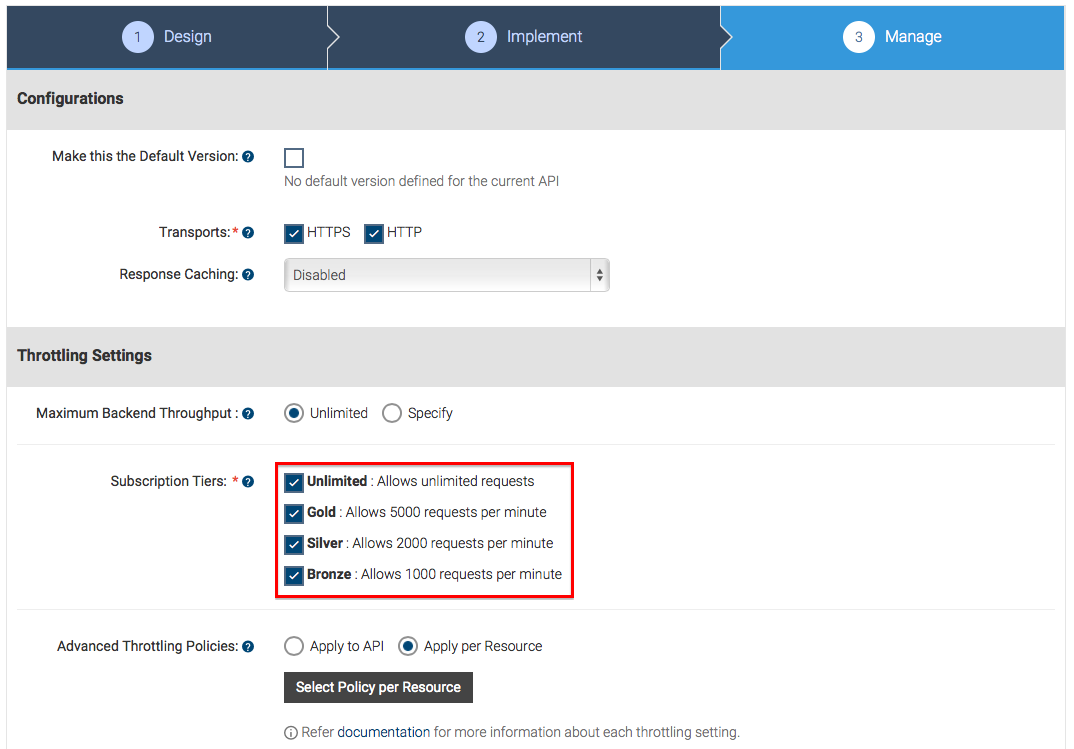
In the next Screen select Managed API and End point Type .

Product and Sandbox Endpoints should point to the actual REST URL of the Product service.



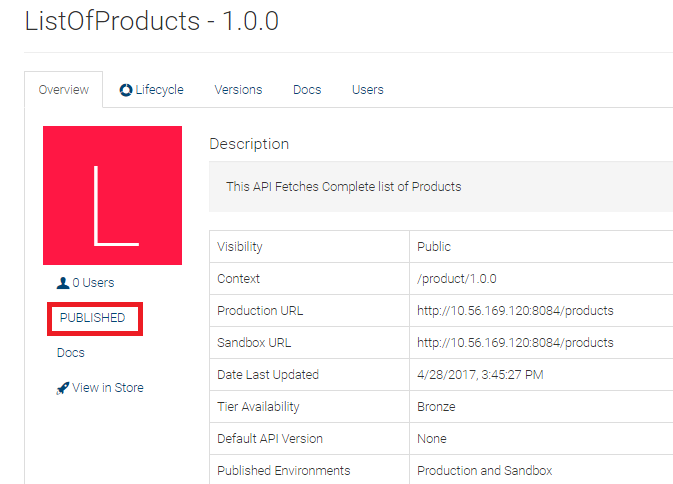
8. Click **Next: Manage >** and enter the information in the table below.

| **Field** | **Sample value** | **Description** |
| --- | --- | --- |
| Transports | HTTP and HTTPS | The transport protocol on which the API is exposed. Both HTTP and HTTPS transports are selected by default. If you want to limit API availability to only one transport (e.g., HTTPS), un-check the other transport. |
| Subscription Tiers | Select all | The API can be available at different levels of service. They allow you to limit the number of successful hits to an API during a given period of time. |



Click **Save & Publish**.   
This publishes the API that you just created to the API Store so that subscribers can use it.

You have created an API.

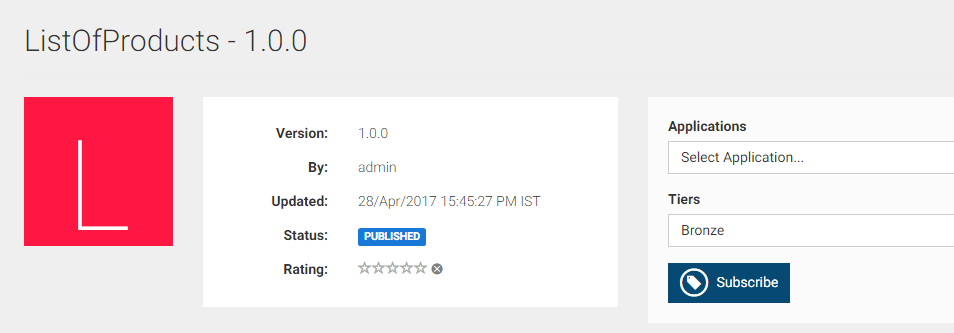


# [Subscribe to an API](https://docs.wso2.com/display/AM210/Subscribe+to+an+API)

1.Sign in to the WSO2 API Store (https://<hostname>:9443/store) and click on an API (e.g., PhoneVerification 1.0.0) to open it.

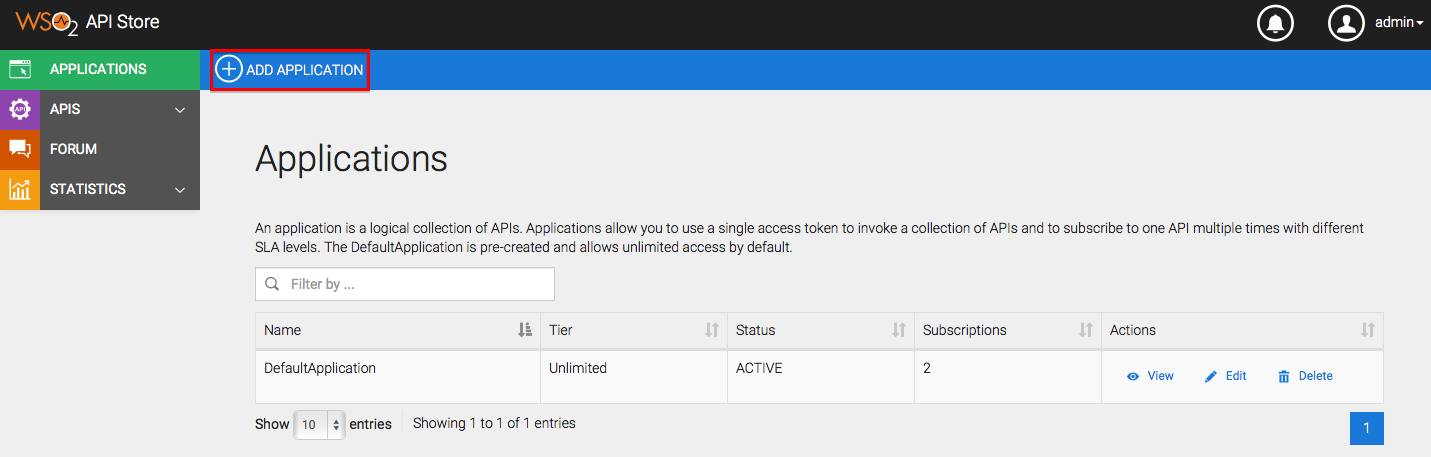
In a multi-tenanted WSO2 API Manager setup, you can access any tenant's store using the URL http://<hostname>/Store?tenant=<tenant\_name>.

2.Note the subscription options for the REST API.



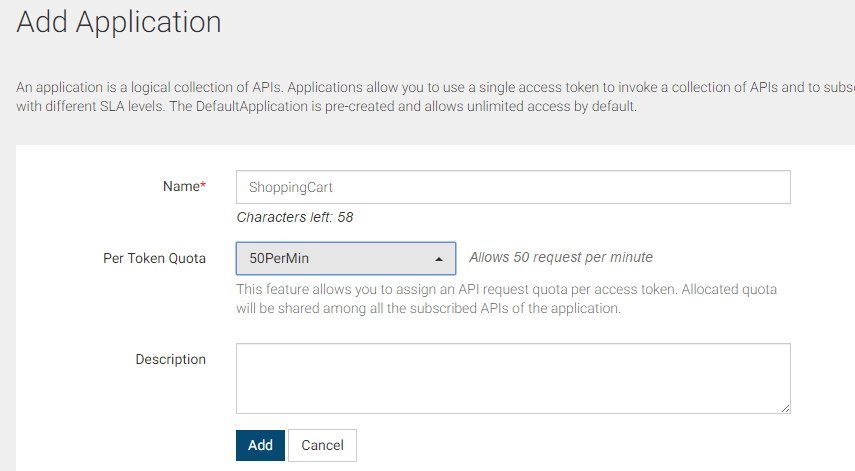
**Create Application**

3.Click the**Applications** menu and click **Add Application** to create a new application.



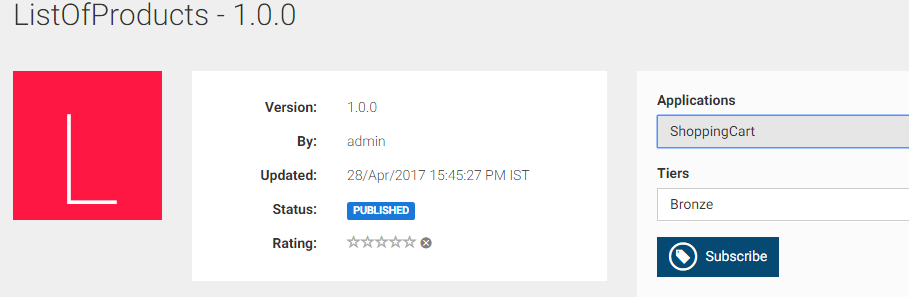
NOTE : Instead of creating a new application, you can also use the default application.

4.Enter the name as TestApp and select the per token quota as 50PerMin for the application and click **Add**.



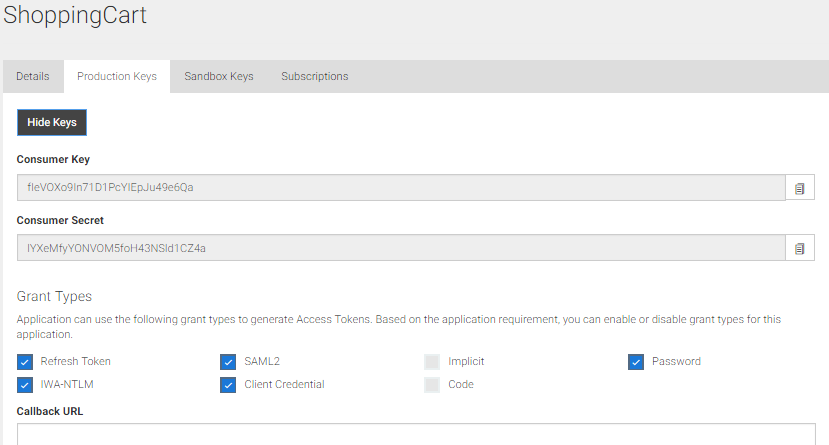
5.Click **APIs** and click on the ListOfProducts API to view the API's subscription options.

6.Select the application that you just created, a tier, and click **Subscribe**.



7.Click the **View Subscriptions** button when prompted. The **Subscriptions** tab opens.

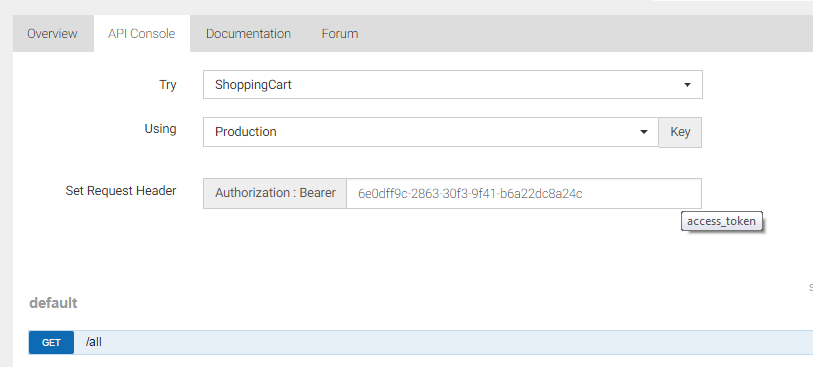
8.Click the **Production Keys** tab and click **Generate Keys** to create an application access token. You can use this token to invoke all APIs that you subscribe to using the same application



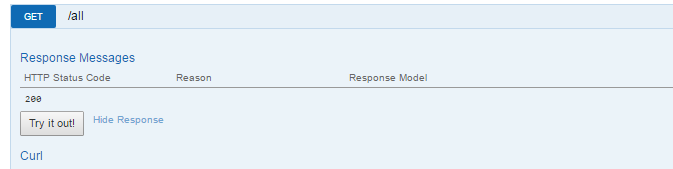
NOTE: You can set a token validity period in the **Access token validity period** text box. By default, it is set to one hour. If you set a minus value (e.g., -1), the token never expires.

# Invoking API using API console

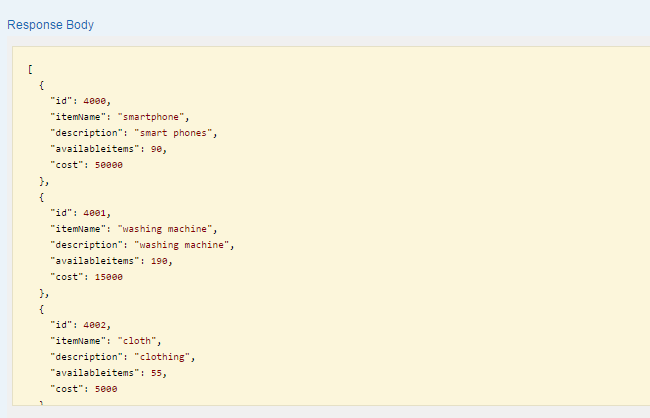
1.Click the **APIs** menu in the API Store and then click on the API that you want to invoke. When the API opens, go to its **API Console** tab



2.Expand the GET method of the resource **all** and click **Try it out** to invoke the API.



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



You have invoked an API using the API Console Sucessfully.

Wso2 APIM Analytics

The steps below explain how to configure WSO2 API Manager Analytics with the API Manager.

Installation Steps:

1. Download the WSO2 APIM Analytics distribution by clicking **ANALYTICS** in the [WSO2 API Management page](http://wso2.com/api-management/). It is best to download and extract it to the same directory to which you downloaded WSO2 API Manager.
2. If you have the API Manager server running, stop the server.
3. If you are running on Windows, download the snappy-java\_1.1.1.7.jar from [here](http://mvnrepository.com/artifact/org.xerial.snappy/snappy-java/1.1.1.7) and copy the JAR file to the <ANALYTICS\_HOME>\repository\components\lib directory.

# [Configuring APIM Analytics](https://docs.wso2.com/display/AM210/Configuring+APIM+Analytics) with API Manager

The API Manager integrates with the [WSO2 Analytics platform](http://wso2.com/analytics) to provide reports, statistics and graphs on the APIs deployed in WSO2 API Manager. You can then configure alerts to monitor these APIs and detect unusual activity, manage locations via geo location statistics and carry out detailed analysis of the logs.

NOTE : By default, WSO2 API Manager has a port offset of 0 (no port offset) and WSO2 API Manager Analytics has an offset of 1. Therefore, this guide assumes that you do not have any other carbon servers running on the same machine with port offsets of 0 or 1.

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

<Enabled>true</Enabled>

2.Share the WSO2AM\_STATS\_DB datasource between WSO2 APIM and WSO2 APIM Analytics as follows.

Open the <API-M\_HOME>/repository/conf/datasources/master-datasources.xml file and make sure that a configuration for the WSO2AM\_STATS\_DB datasource is included. The default configuration is as follows.

<datasource>

    <name>WSO2AM\_STATS\_DB</name>

    <description>The datasource used for getting statistics to API Manager</description>

    <jndiConfig>

        <name>jdbc/WSO2AM\_STATS\_DB</name>

    </jndiConfig>

    <definition type="RDBMS">

        <configuration>

            <url>jdbc:h2:../tmpStatDB/WSO2AM\_STATS\_DB;DB\_CLOSE\_ON\_EXIT=FALSE;LOCK\_TIMEOUT=60000;AUTO\_SERVER=TRUE</url>

            <username>wso2carbon</username>

            <password>wso2carbon</password>

            <defaultAutoCommit>false</defaultAutoCommit>

            <driverClassName>org.h2.Driver</driverClassName>

            <maxActive>50</maxActive>

            <maxWait>60000</maxWait>

            <testOnBorrow>true</testOnBorrow>

            <validationQuery>SELECT 1</validationQuery>

            <validationInterval>30000</validationInterval>

        </configuration>

    </definition>

</datasource>

3.Open the <AM\_ANALYTICS\_HOME>/repository/conf/datasources/stats-datasources.xml file and make sure that the same configuration in the

<API-M\_HOME>/repository/conf/datasources/master-datasources.xml file (mentioned in the previous sub step) is added in it.

Open the <API-M\_HOME>/repository/conf/log4j.properties file. Add the DAS\_AGENT to the end of the root logger as shown below.

2.Open the <APIM\_HOME>/repository/conf/log4j.properties file. Add DAS\_AGENT to the end of the log4j.rootLogger property  as shown in the example below.

log4j.rootLogger=ERROR, CARBON\_CONSOLE, CARBON\_LOGFILE, CARBON\_MEMORY, CARBON\_SYS\_LOG, ERROR\_LOGFILE, DAS\_AGENT

NOTE :This configuration is required only if you want to analyze WSO2 APIM logs using the Log Analyzer.

|  |
| --- |
|  |

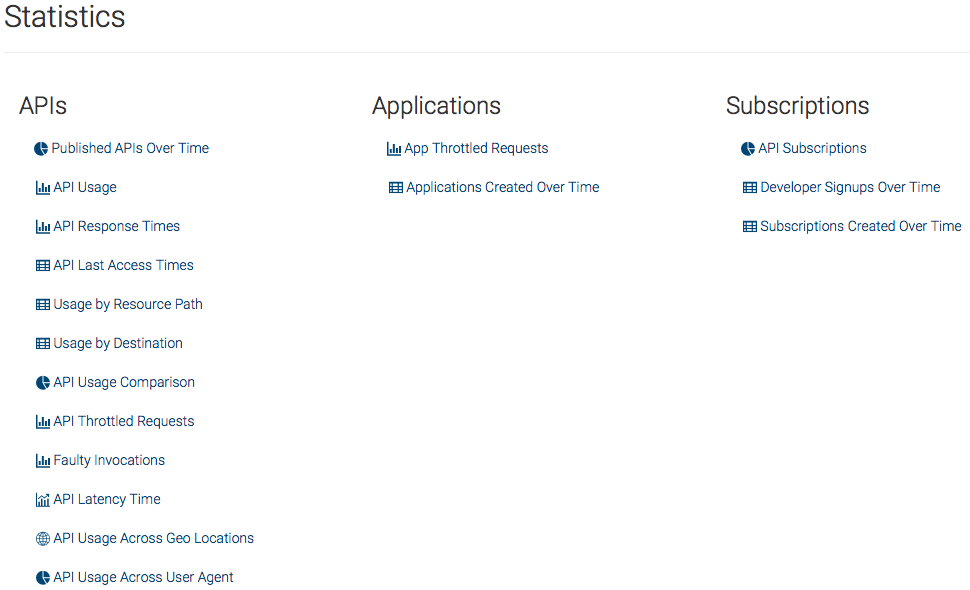
4.Start the WSO2 APIM Analytics server, and then start the API Manager server. To start a WSO2 product server, navigate to the <PRODUCT\_HOME>/bin directory in your console and run one of the following scripts as relevant.

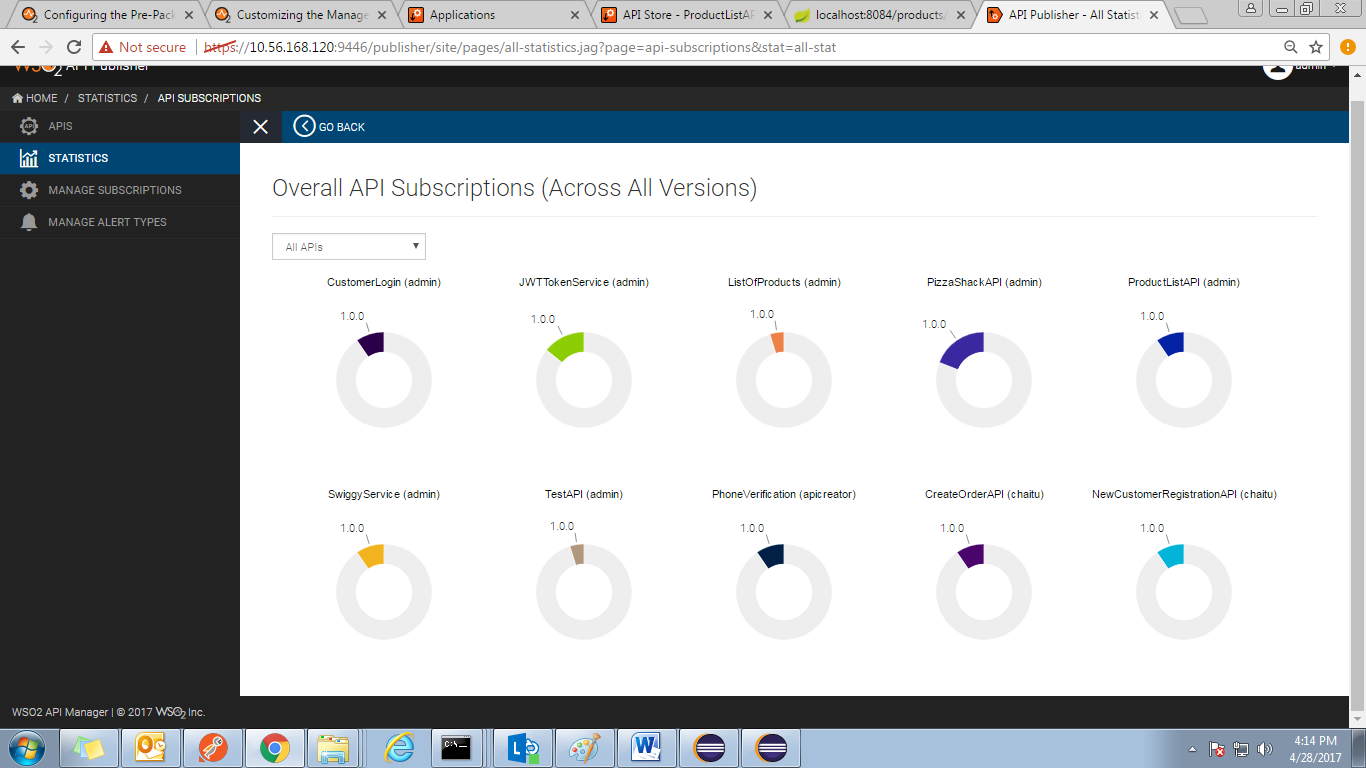
* + 1. On Windows:  wso2server.bat --run
    2. On Linux/Mac OS:  sh wso2server.sh

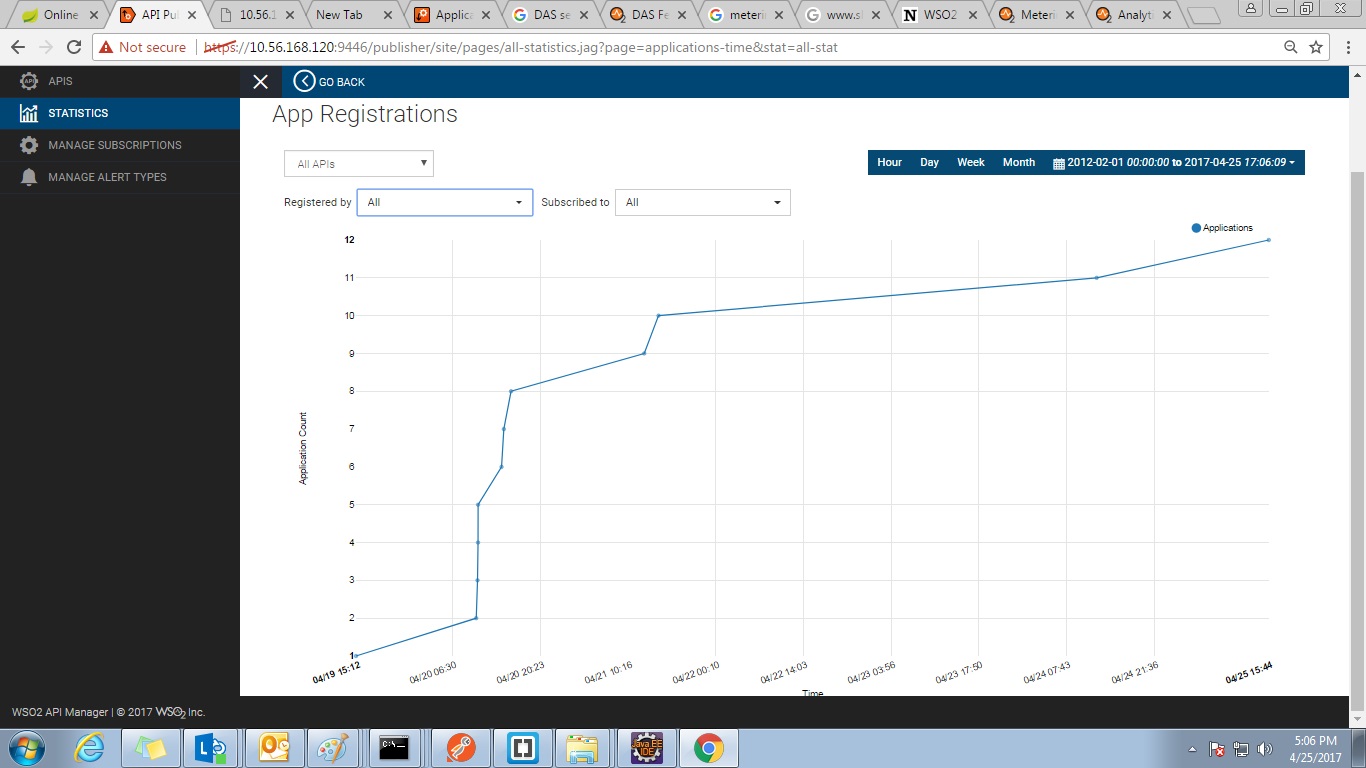
4.Invoke several APIs to generate some statistical data and wait a few seconds.

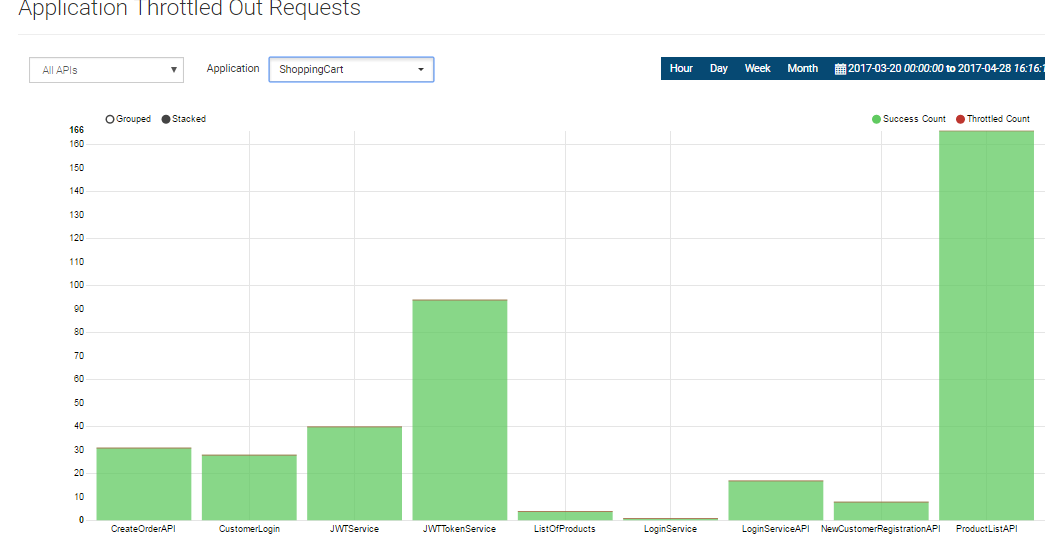
**Statistics Report**

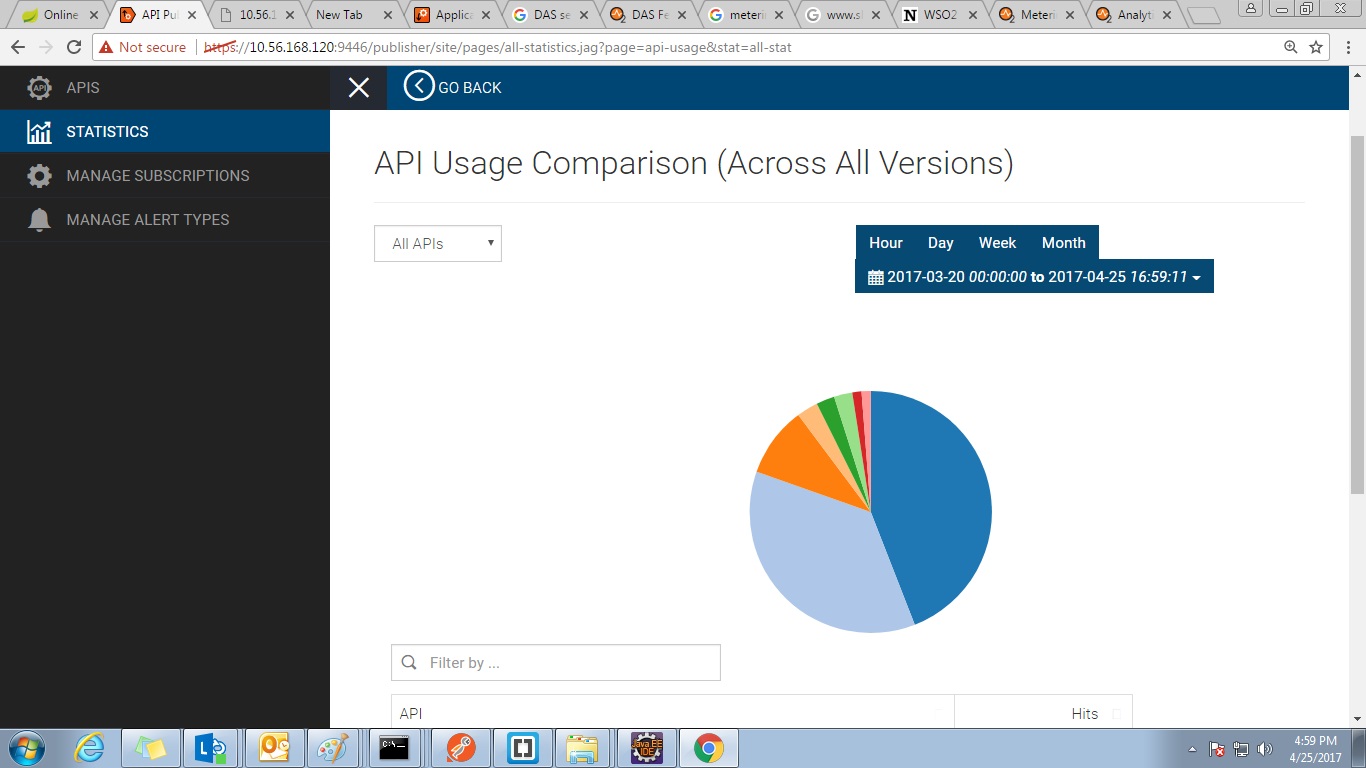
5. Connect to the API Publisher as a creator and click one of the statistical dashboards available in the **Statistics** menu.

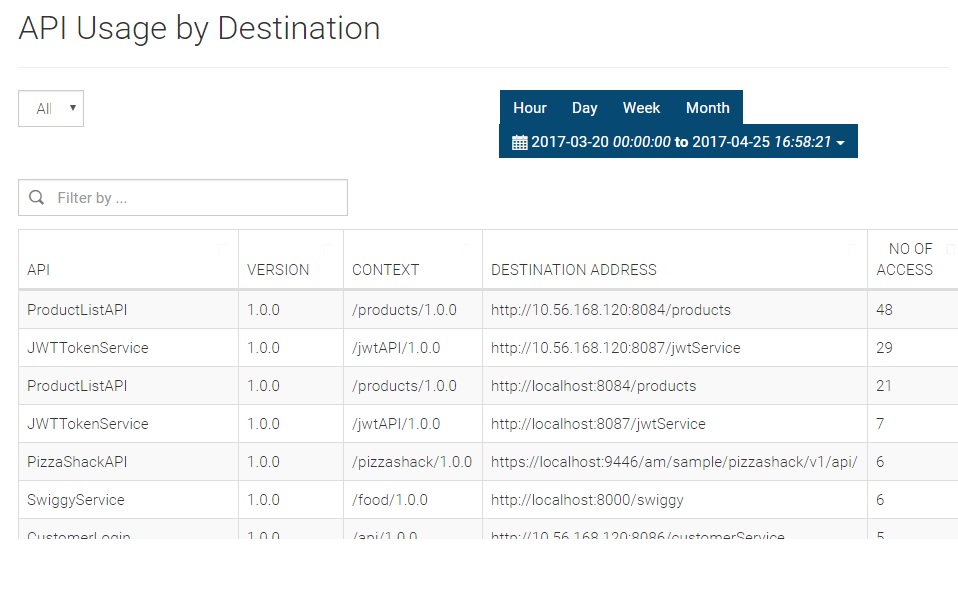


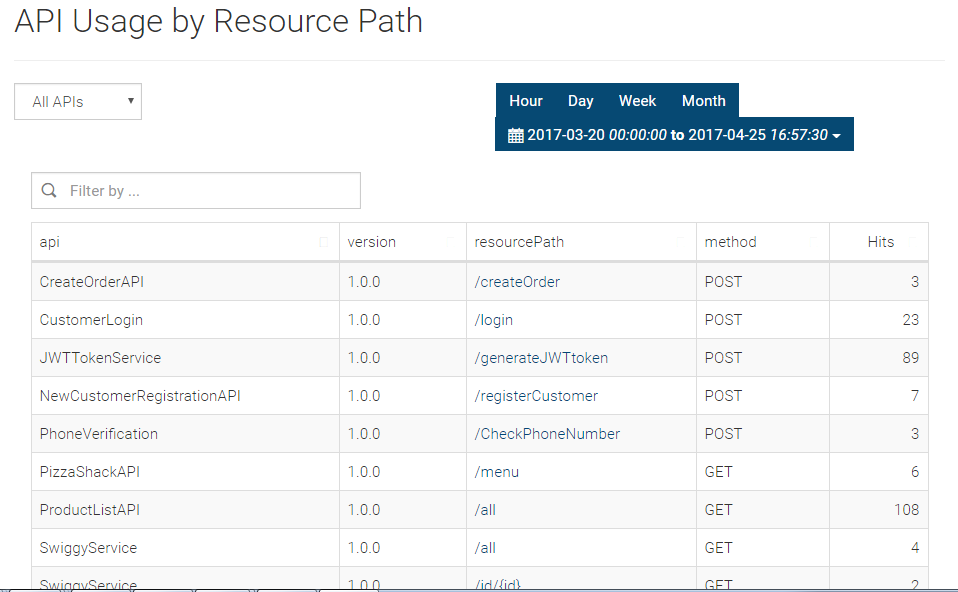


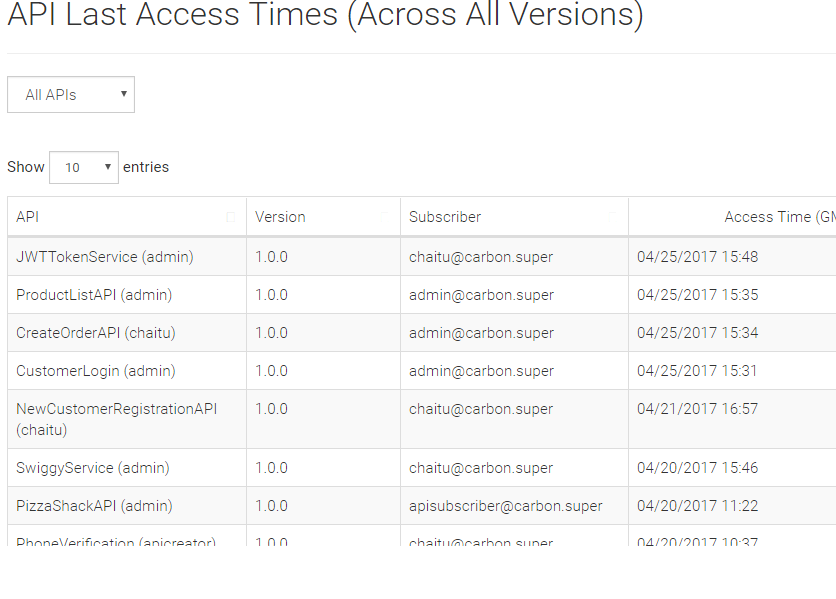


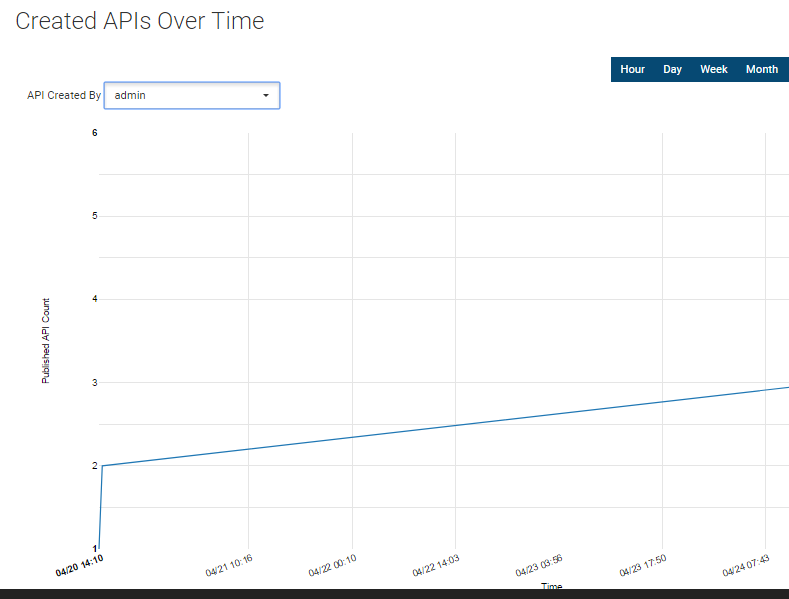


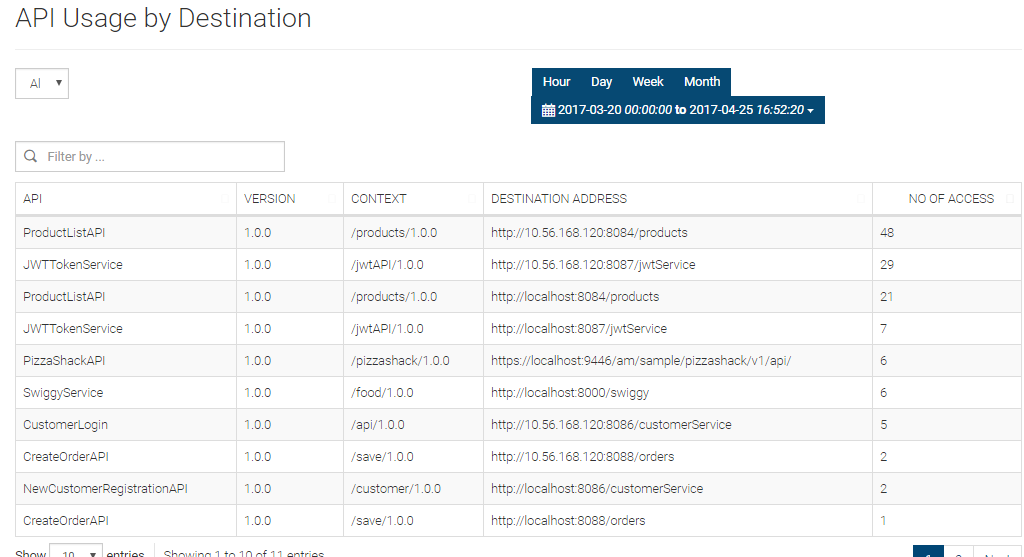


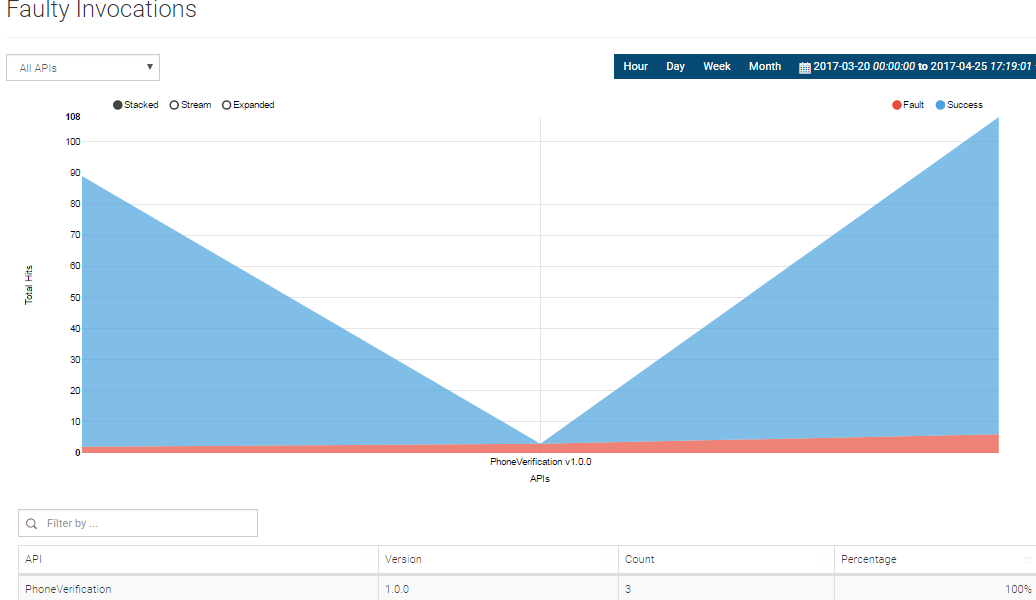


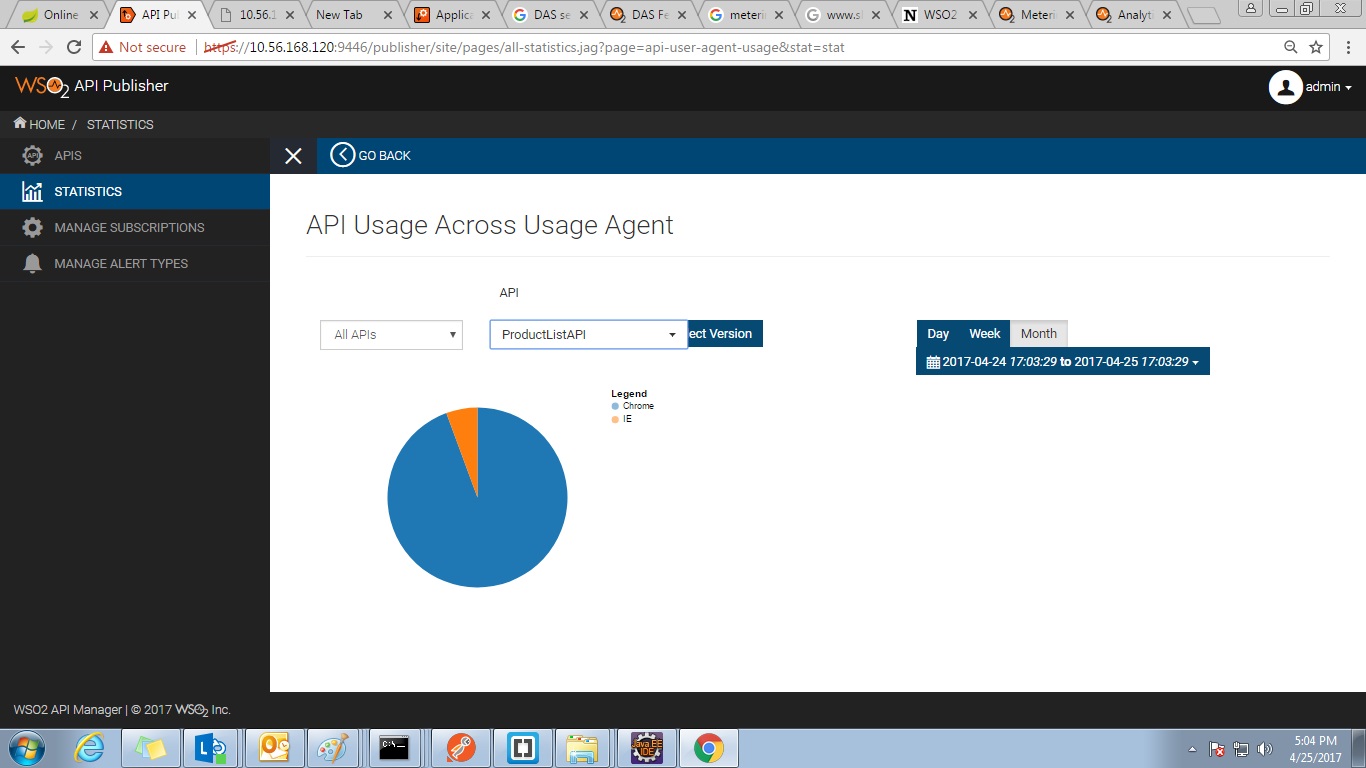


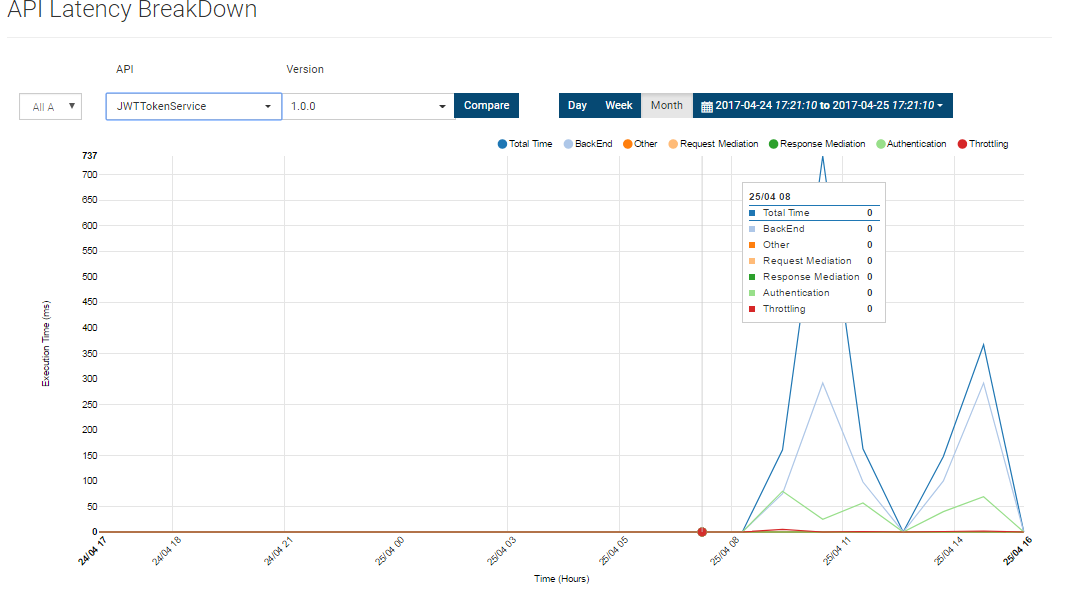


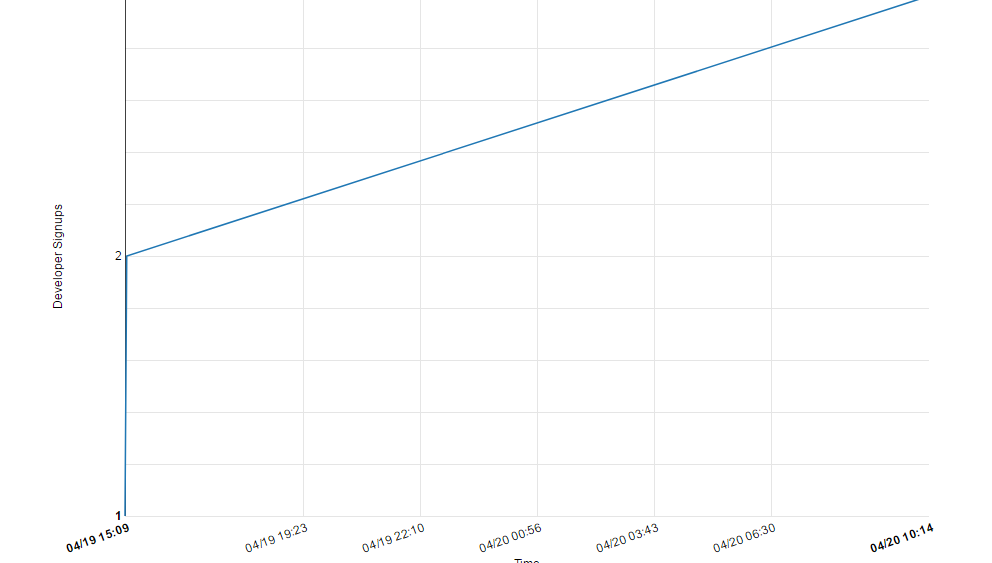












WSo2 Identity Server

WSO2 Identity Server is a product built on top of WSO2 Carbon. Based on the OSGi specification, it enables easy customization and extension through its componentized architecture. This topic describes the architecture of the Identity Server. The users are given the choice of deployment to on-premise servers, private cloud or public cloud without configuration changes.

Each server in the WSO2 platform is built using the Carbon platform. "Carbon server" is a term used to depict any product, such as WSO2 Enterprise Service Bus, WSO2 Application Server, and WSO2 Identity Server, that is built on top of the Carbon platform.

The WSO2 Identity Server is used directly by multiple users, through its user-friendly [Management Console](https://docs.wso2.com/display/IS520/Getting+Started+with+the+Management+Console). Apart from the default admin user (with the user name ‘admin’), other users can be created later by the admin users that have the privileges to create a new user, or by signing up. Each user can have roles, where each role can have privileges assigned to them. A user’s roles can be changed at any time by the admin user. Apart from such registered users, Identity Server is also used as an identity provider for third party applications, which also have their own sets of users.

# [Configuring the Identity Server 5.2.0 with API Manager 2.0.0](https://docs.wso2.com/display/CLUSTER44x/Configuring+the+Identity+Server+5.2.0+as+a+Key+Manager+with+API+Manager+2.0.0)

**Identity Server Configurations**

1. Download the latest version of the Identity Server from

<http://wso2.com/products/identity-server/>

1. Extract the archive file to a dedicated directory for the Identity Server, which will hereafter be referred to as <IS\_HOME>

3.Make the following changes in the **api-manager.xml** file

Change the GatewayType property to the following. This is done because the default value here is Synapse. Synapse runtime is used for various ESB related functionality that is not available in the Identity Server, so this must be changed to None.

<GatewayType>None</GatewayType>

Change EnableThriftServer to false. The Identity Server does not come with a thrift server and this causes issues at runtime if not disabled.

<EnableThriftServer>false</EnableThriftServer>

4.Open the <IS\_HOME>/repository/conf/datasources/master-datasources.xml file and add the following datasources.

Ensure that you keep the 'WSO2\_CARBON\_DB' datasource the way it is and simply add the following datasources in the master-datasources.xml file. Also, note that the WSO2AM\_DB is already added in the master-datasources.xml file so you do not need to add it again. However, you must edit this datasource to point to your new database as this still point to the default H2 database. The following code block includes a sample of the WSO2AM\_DB datasource as a sample configuration when pointing to the new database.

<datasource>

    <name>WSO2AM\_DB</name>

#### <description>The datasource used for API Manager database</description>

    <jndiConfig>

        <name>jdbc/WSO2AM\_DB</name>

    </jndiConfig>

    <definition type="RDBMS">

        <configuration>  <url>jdbc:mysql://localhost:3306/apimgt?autoReconnect=true&amp;relaxAutoCommit=true&amp;</url>

            <username>apiuser</username>

            <password>apimanager</password>

            <driverClassName>com.mysql.jdbc.Driver</driverClassName>

            <maxActive>50</maxActive>

            <maxWait>60000</maxWait>

            <testOnBorrow>true</testOnBorrow>

            <validationQuery>SELECT 1</validationQuery>

            <validationInterval>30000</validationInterval>

            <defaultAutoCommit>false</defaultAutoCommit>

        </configuration>

    </definition>

</datasource>

<datasource>

    <name>WSO2REG\_DB</name>

    <description>The datasource used for registry</description>

    <jndiConfig>

        <name>jdbc/WSO2REG\_DB</name>

    </jndiConfig>

    <definition type="RDBMS">

   <configuration>

<url>jdbc:mysql://localhost:3306/registry?autoReconnect=true&relaxAutoCommit=true&amp</url>

            <username>apiuser</username>

            <password>apimanager</password>

            <driverClassName>com.mysql.jdbc.Driver</driverClassName>

            <maxActive>50</maxActive>

            <maxWait>60000</maxWait>

            <testOnBorrow>true</testOnBorrow>

            <validationQuery>SELECT 1</validationQuery>

            <validationInterval>30000</validationInterval>

        </configuration>

    </definition>

</datasource>

<datasource>

    <name>WSO2UM\_DB</name>

    <description>The datasource used for user management</description>

    <jndiConfig>

        <name>jdbc/WSO2UM\_DB</name>

    </jndiConfig>

    <definition type="RDBMS">

        <configuration>

   <url>jdbc:mysql://localhost:3306/userstore?autoReconnect=true&amp;relaxAutoCommit=true&amp;

            </url>

            <username>apiuser</username>

            <password>apimanager</password>

            <driverClassName>com.mysql.jdbc.Driver</driverClassName>

            <maxActive>50</maxActive>

            <maxWait>60000</maxWait>

            <testOnBorrow>true</testOnBorrow>

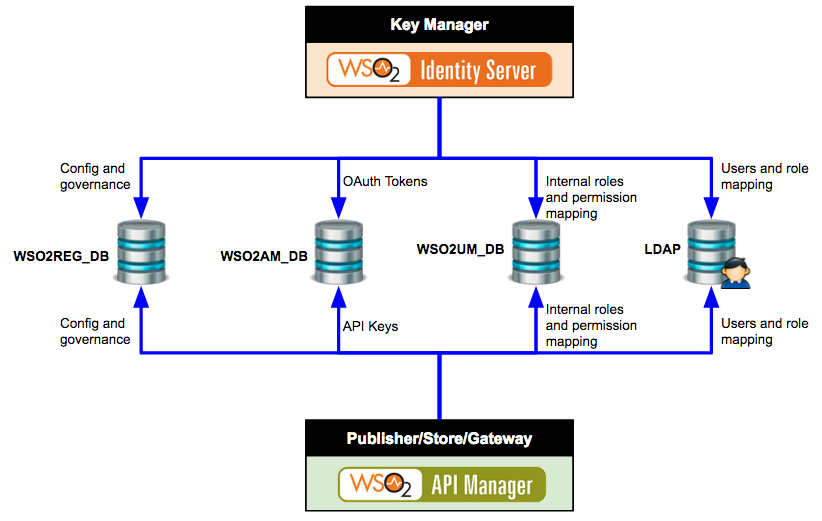
            <validationQuery>SELECT 1</validationQuery>

            <validationInterval>30000</validationInterval>

        </configuration> </definition>

</datasource>

5.The following diagram illustrates how databases are shared between IS and APIM as per the above configuration.



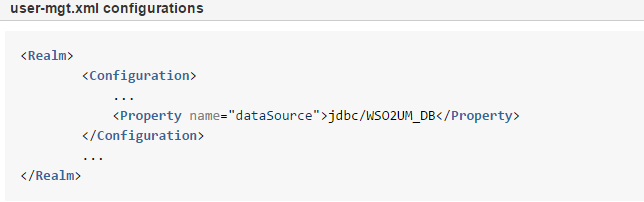
* **WSO2REG\_DB** - This is used to keep the registry information.
* **WSO2UM\_DB** - This is used to store the permissions (i.e. permission store) and the internal roles of the users.
* **WSO2AM\_DB** - This will be used to keep the identity data and API-related data. This includes OAuth tokens and keys. When serving key-validation requests, the key manager validates whether there are subscriptions made by the particular key. For this WSO2AM\_DB should be accessed.

When doing this change, do not replace the existing <dbConfig> for "wso2registry". Simply add the following configuration to the existing configurations.

6.Make the following change to the  <IS\_HOME>/repository/conf/registry.xml file. Create the registry mounts by inserting the following sections into the **registry.xml** file.



7.Change the datasource in the **user-mgt.xml** file found in the <IS\_HOME>/repository/conf/ directory to WSO2UM\_DB.



8.Add the user store configuration correctly in the <IS\_HOME>/repository/conf/user-mgt.xml file so that both the Identity Server and API Manager point to the same user store

You must change the <UserStoreManager> element here since the internal LDAP user store is used by default. The <UserStoreManager class="org.wso2.carbon.user.core.ldap.ReadWriteLDAPUserStoreManager"> code block needs to be removed or modified and the right code block must be used.

JWT configuration must be done in the <IS\_HOME>/repository/conf/api-manager.xml file in the Identity Server.

Enable the ClaimsRetrieverImplClass, ConsumerDialectURI and SignatureAlgorithm. Set SignatureAlgorithm to NONE.

**DataBase Creation**

Create the following databases in the MySQL database server.

* Userstore
* Registry
* apimgt

For creating the userstore and registry database, use the <IS\_HOME>/dbscripts/mysql.sql script.

When creating the apimgt db, run the following script; <APIM\_HOME>/dbscripts/apimgt/mysql.sql. The script found in the <APIM\_HOME>/dbscripts/apimgt/ directory has all the tables required to manage OAuth access tokens and also includes other identity-related features

Create a user ‘apiuser’ with password ‘apimanager’. Grant all permissions for this user in the above three databases. For example:

grant all on apimgt.\* TO apiuser@localhost identified by "apimanager";

grant all on userstore.\* TO apiuser@localhost identified by "apimanager";

grant all on registry.\* TO apiuser@localhost identified by "apimanager";

### Configuring the API Manager

1. Download WSO2 API Manager from [here](http://wso2.com/products/api-manager/) and [install it](https://docs.wso2.com/display/AM200/Installing+the+Product).
2. Open the <APIM\_HOME>/repository/conf/datasources/master-datasources.xml file and add the following datasources.

<datasource>

<name>WSO2AM\_DB</name>

<description>The datasource used for API Manager database</description>

<jndiConfig>

<name>jdbc/WSO2AM\_DB</name>

</jndiConfig>

<definition type="RDBMS">

<configuration> <url>jdbc:mysql://localhost:3306/apimgt?autoReconnect=true&amp;relaxAutoCommit=true&amp;</url>

<username>apiuser</username>

<password>apimanager</password>

<driverClassName>com.mysql.jdbc.Driver</driverClassName>

<maxActive>50</maxActive>

<maxWait>60000</maxWait>

<testOnBorrow>true</testOnBorrow>

<validationQuery>SELECT 1</validationQuery>

<validationInterval>30000</validationInterval>

<defaultAutoCommit>false</defaultAutoCommit>

</configuration>

</definition>

</datasource>

<datasource>

<name>WSO2REG\_DB</name>

<description>The datasource used for registry and user manager</description>

<jndiConfig>

<name>jdbc/WSO2REG\_DB</name>

</jndiConfig>

<definition type="RDBMS">

<configuration>

<url>jdbc:mysql://localhost:3306/registry?autoReconnect=true&amp;relaxAutoCommit=true&amp;</url>

<username>apiuser</username>

<password>apimanager</password>

<driverClassName>com.mysql.jdbc.Driver</driverClassName>

<maxActive>50</maxActive>

<maxWait>60000</maxWait>

<testOnBorrow>true</testOnBorrow>

<validationQuery>SELECT 1</validationQuery>

<validationInterval>30000</validationInterval>

</configuration>

</definition>

</datasource>

<datasource>

<name>WSO2UM\_DB</name>

<description>The datasource used for registry and user manager</description>

<jndiConfig>

<name>jdbc/WSO2UM\_DB</name>

</jndiConfig>

<definition type="RDBMS">

<configuration>

<url>jdbc:mysql://localhost:3306/userstore?autoReconnect=true&amp;relaxAutoCommit=true&amp;

</url>

<username>apiuser</username>

<password>apimanager</password>

<driverClassName>com.mysql.jdbc.Driver</driverClassName>

<maxActive>50</maxActive>

<maxWait>60000</maxWait>

<testOnBorrow>true</testOnBorrow>

<validationQuery>SELECT 1</validationQuery>

<validationInterval>30000</validationInterval>

</configuration>

</definition>

</datasource>

3. Open the **user-mgt.xml** file found in the <APIM\_HOME>/repository/conf directory and change the permission datasource.

a. Add the datasource configuration as below

**<Realm>**

**<Configuration>**

**...**

**<Property name="dataSource">jdbc/WSO2UM\_DB</Property>**

**</Configuration>**

**...**

**</Realm>**

b.Configure the <UserStoreManager> section of the <AM\_HOME>/repository/conf/user-mgt.xml file of the API Manager .

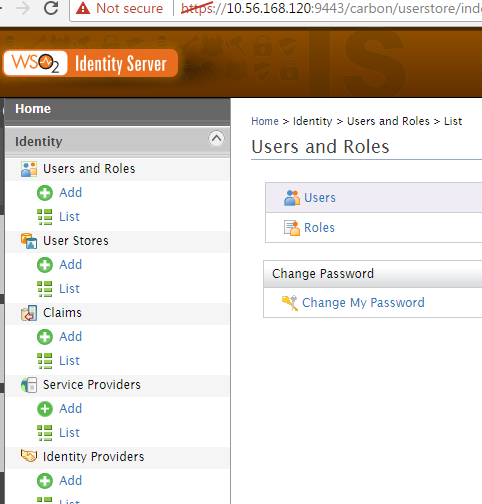
**NOTE: Make sure you add the user store configuration correctly. This is the same configuration that you did in the Identity Server**

4.Create the registry mounts. Open the <APIM\_HOME>/repository/conf/registry.xml file and insert the following sections.



**NOTE :Make sure you add the MySQL JDBC driver to both servers. I.e. put the .jar file into the <PRODUCT\_HOME>/repository/components/lib directory.**

Start the Identity Server . and access the console https://<ipaddress>:<port>/carbon.

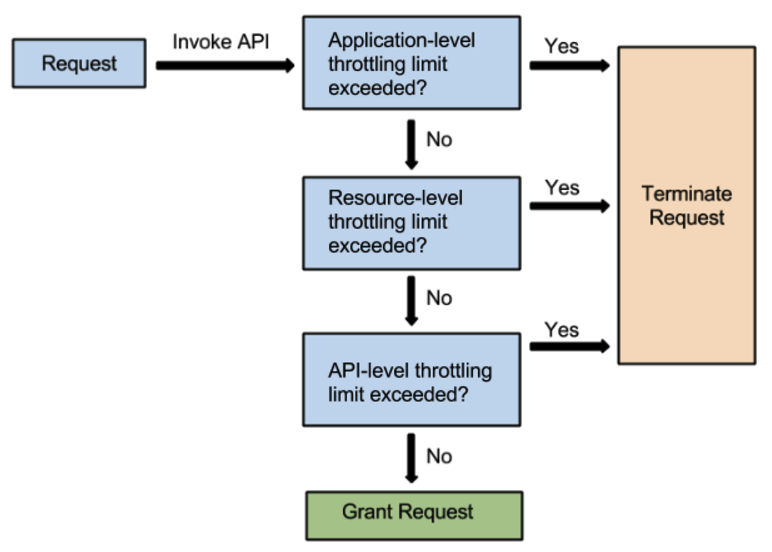


# [Wso2 Throttling](https://docs.wso2.com/display/AM1100/Introducing+Throttling)

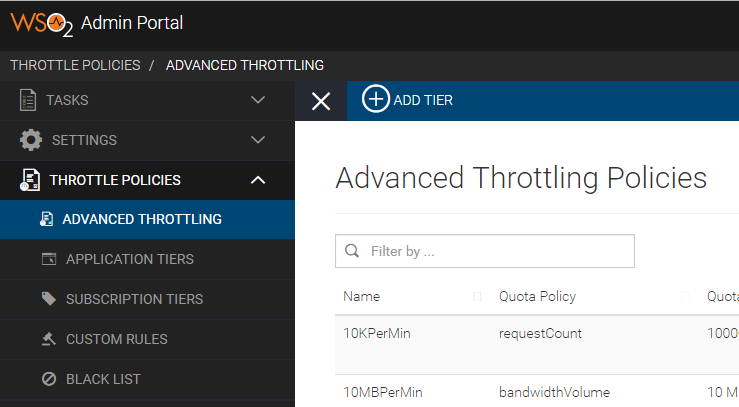
Throttling allows you to limit the number of successful hits to an API during a given period of time, typically in cases such as the following:

* To protect your APIs from common types of security attacks such as denial of service (DOS)
* To regulate traffic according to infrastructure availability
* To make an API, application or a resource available to a consumer at different levels of service, usually for monetization purpose

You can define throttling in the API, application and resource levels. The final request limit granted to a given user on a given API is ultimately defined by the consolidated output of all throttling tiers together.



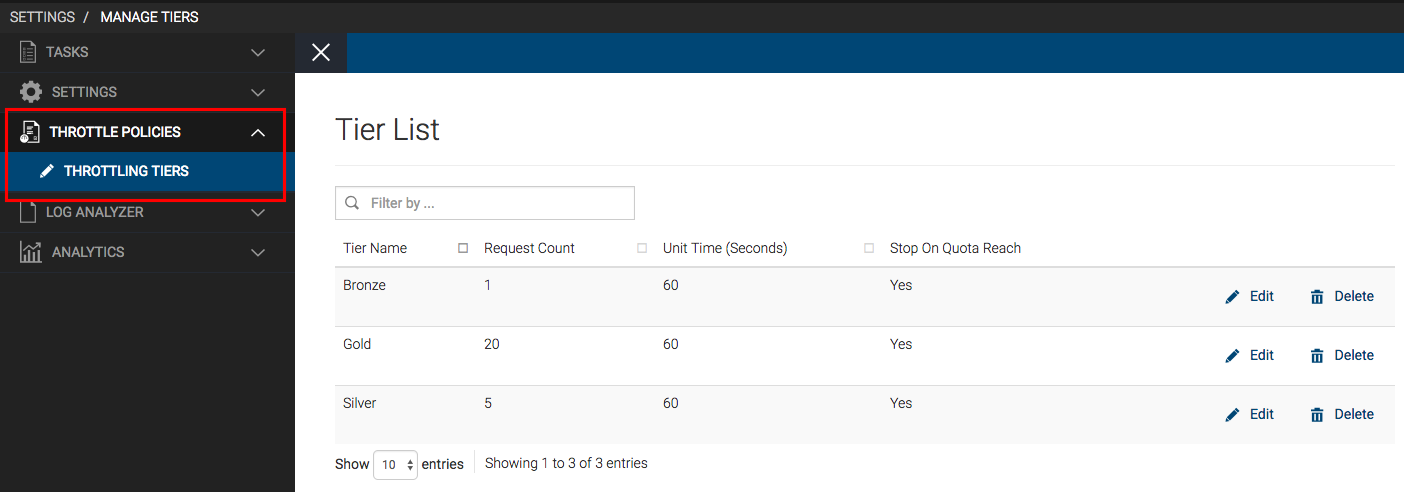
Throttling can be accessed using admin portal of API Manager. https://<Ipaddress>:<port>/admin

[Adding New Throttling Policies](https://docs.wso2.com/display/AM200/Adding+New+Throttling+Policies)

API Manager admins can add new throttling policies and define extra properties to the throttling policies. Click the level of throttling that you want to add a new policy in to see instructions:

* [Adding a new advanced throttling policy](https://docs.wso2.com/display/AM200/Adding+New+Throttling+Policies#AddingNewThrottlingPolicies-Addinganewadvancedthrottlingpolicy)
* [Adding a new application-level throttling tier](https://docs.wso2.com/display/AM200/Adding+New+Throttling+Policies#AddingNewThrottlingPolicies-Addinganewapplication-levelthrottlingtier)
* [Adding a new subscription-level throttling tier](https://docs.wso2.com/display/AM200/Adding+New+Throttling+Policies#AddingNewThrottlingPolicies-Addinganewsubscription-levelthrottlingtier)

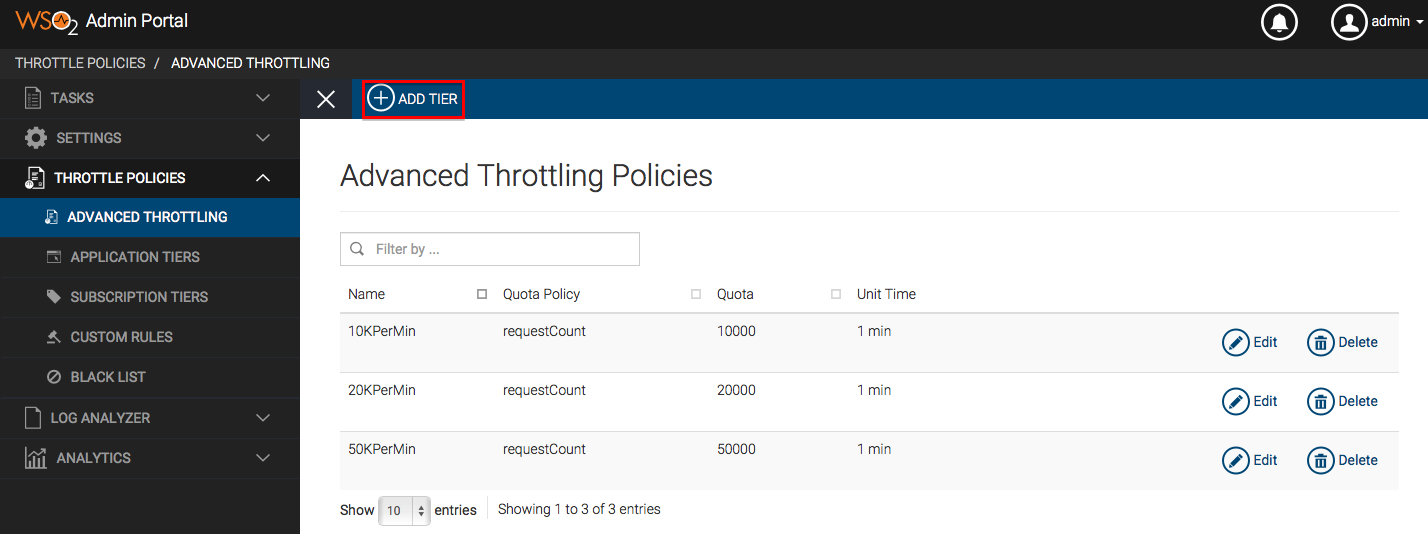
NOTE : To make changes in the throttling configurations, the EnableAdvanceThrottling parameter has to be set to true in api-manager.xml. This is set to true by default. If you change it to false, you will only be able to see the available tiers.



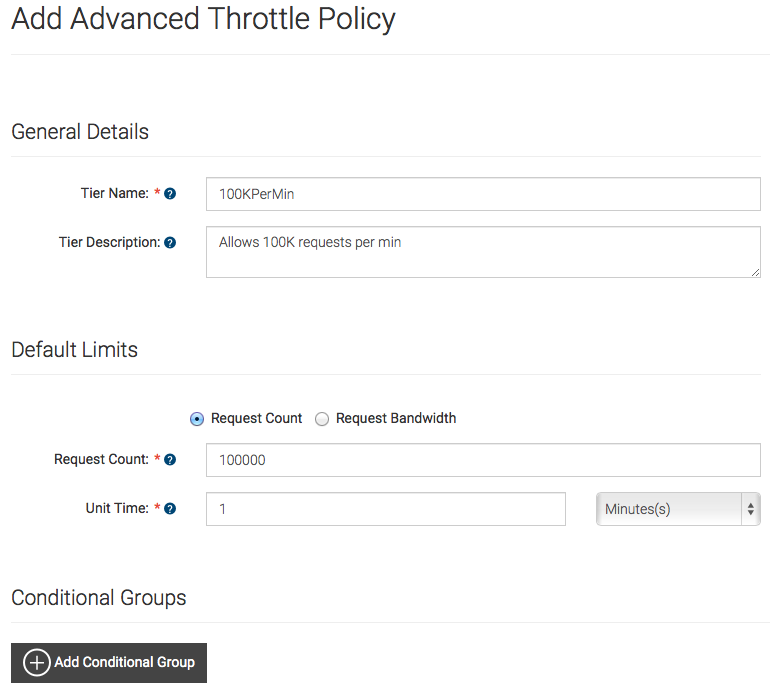
### Adding a new advanced throttling policy

Advanced throttling policies are applicable to both APIs and resources.

1. Log in to the Admin Portal using the URL https://localhost:9443/admin and your admin credentials.
2. Click **Advanced Throttling** under the **Throttle Policies** section. The existing set of throttling tiers are displayed. To add a new tier, click **Add Tier**.



Fill in the details required by this form and click **Save** once you are done.



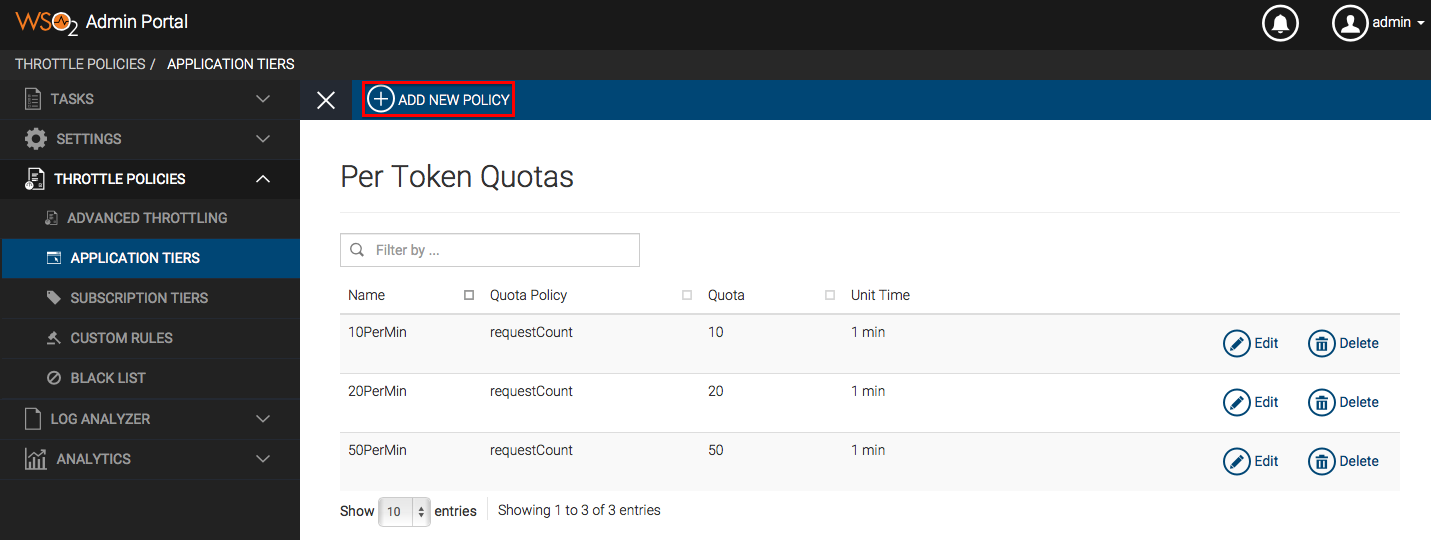
If you want to add throttling limits with different parameters to the conditions below, click **Add Conditional Group**.

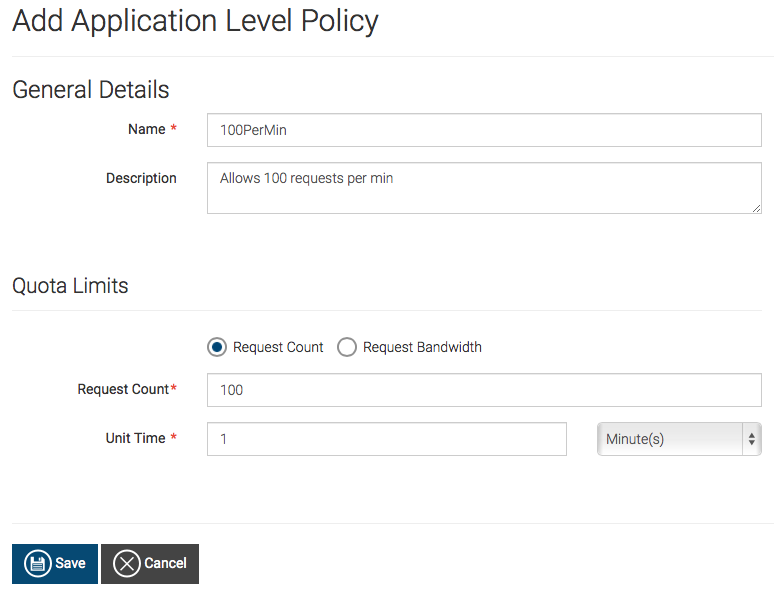
Note that if you want to add a header, query param or JWT claim condition, you need to set the <EnableHeaderConditions>,<EnableJWTClaimConditions> or <EnableQueryParamConditions> element to true (depending on which condition you need) in the repository/conf/api-manager.xml file.

### Adding a new application-level throttling tier

Application-level throttling policies are applicable per access token generated for an application.

1. Log in to the Admin Portal using the URL https://localhost:9443/admin and your admin credentials.
2. Click **Application Tiers** under the **Throttle Policies** section. The existing set of throttling tiers are displayed. To add a new tier, click **Add New Policy**.





### You have added a new application-level throttling policy.

### Adding a new subscription-level throttling tier

1. Log in to the Admin Portal using the URL  https://localhost:9443/admin  and your admin credentials.
2. Click **Subscription Tiers** under the **Throttle Policies** section. The existing set of throttling tiers are displayed. To add a new tier, click **Add Tier**.

### 

### Fill in the details required by this form and click ****Save**** once you are done.

### 

Given below is a description of the fields you find in the form:

| Field | **Description** |
| --- | --- |
| Request Count/Request Bandwidth | The maximum number of requests/maximum bandwidth allowed to the API within the time period given in the next field. |
| Unit Time | Time within which the the number of requests given in the previous field is allowed to the API. This can be defined in minutes, hours, days, weeks, months or years. |
| Burst Control (Rate Limiting) | You can define the request count/bandwidth per unit time on an addition layer by using rate limiting. This is usually a smaller number of requests/bandwidth for a shorter time span than what is enforced in the above fields. For instance, if there's a subscription level policy enforced over a long period, you may not want users to consume the entire quota within a short time span. Enforcing a rate limit protects the backend from sudden request bursts and controls the usage at a subscription and API level. |
| Stop On Quota Reach | This indicates the action to be taken when a user goes beyond the allocated quota. If the check box is selected, the user's requests are dropped and an error response (HTTP Status code 429) is given. If the check box is cleared, the requests are allowed to pass through. |
| Billing Plan | This field only makes sense if you have API Monetization enabled. An API is determined as Free, Commercial or Freemium depending on the tiers it has. If all tiers of an API are tagged as Free, the API will be labelled as a Free API. This labeling happens on the API Store only if monetization has been enabled. |
| Custom Attributes | Custom attribute values are displayed as key value pairs on the API Store's API subscription page. The main objective of these fields are to provide more information regarding the tier to Application Developers at the time of API subscription. |
| Permissions | You can allow or deny permission for specific roles. |

You have added a new subscription-level throttling policy.