Quick Start Guide

WSO2 API Manager is a complete solution for publishing APIs, creating and managing a developer community and for routing API traffic in a scalable manner. It leverages the integration, security and governance components from the WSO2 Enterprise Service Bus, WSO2 Identity Server, and WSO2 Governance Registry. In addition, as it is powered by the WSO2 Business Activity Monitor (BAM), the WSO2 API Manager is ready for massively scalable deployments immediately.

Before you begin,

1. Install Oracle Java SE Development Kit (JDK) version 1.6.24 or later or 1.7.* and set the `JAVA_HOME` environment variable.
2. Download 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.)

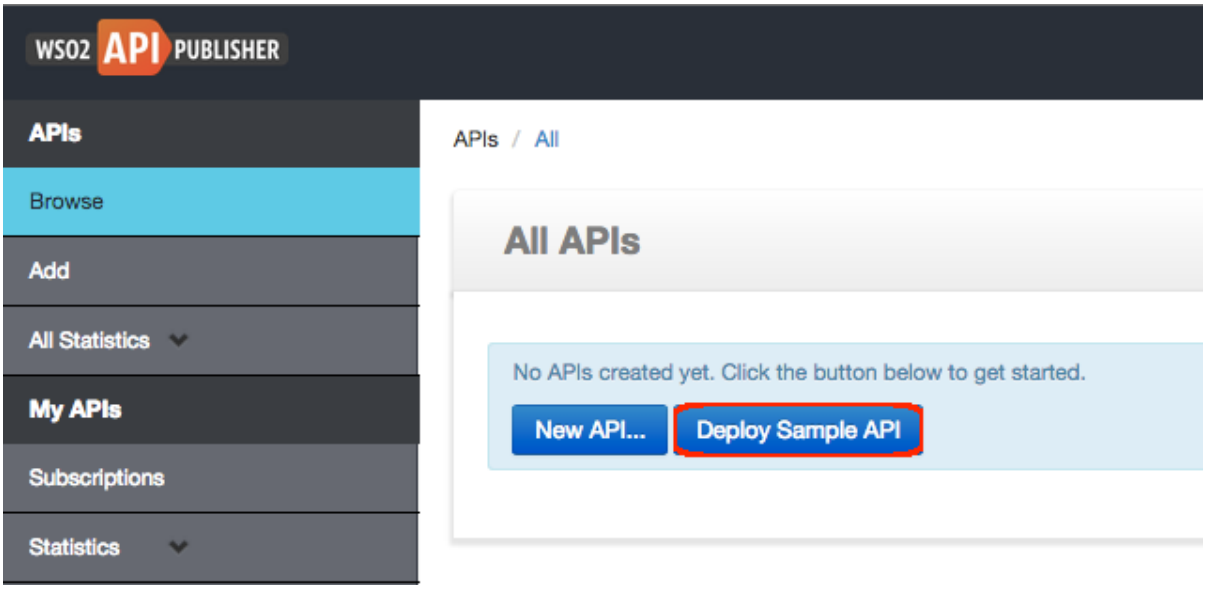
Let's go through the use cases of the API Manager:

- 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
    - Publishing the API
    - Subscribing to the API
    - Invoking the API
    - Monitoring APIs and viewing statistics

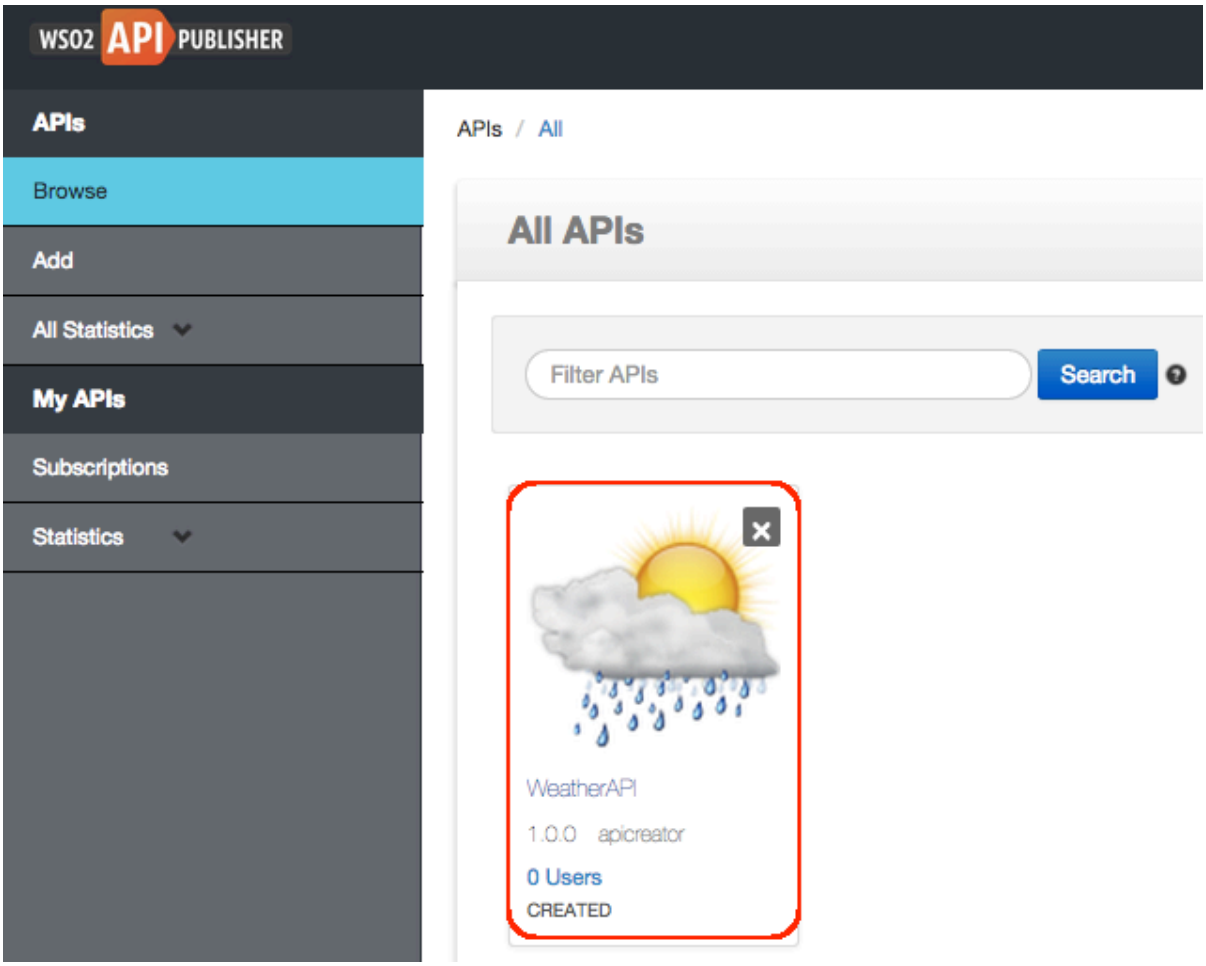
Invoking your first API

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

1. Open the API Publisher (`https://<hostname>:9443/publisher`) and log in with **`admin/admin`** credentials.
2. Click the **Deploy Sample API** button. It deploys a sample API called `WeatherAPI` into the API Manager.

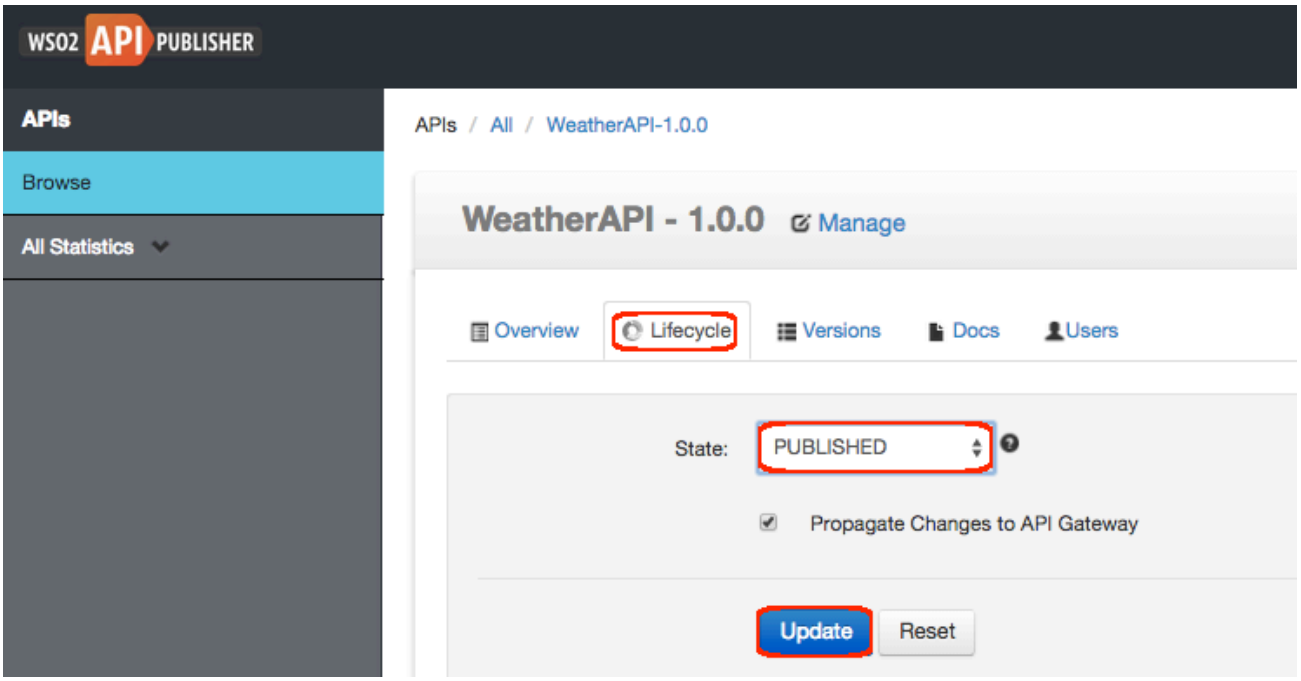


3. Click `WeatherAPI` to open it.

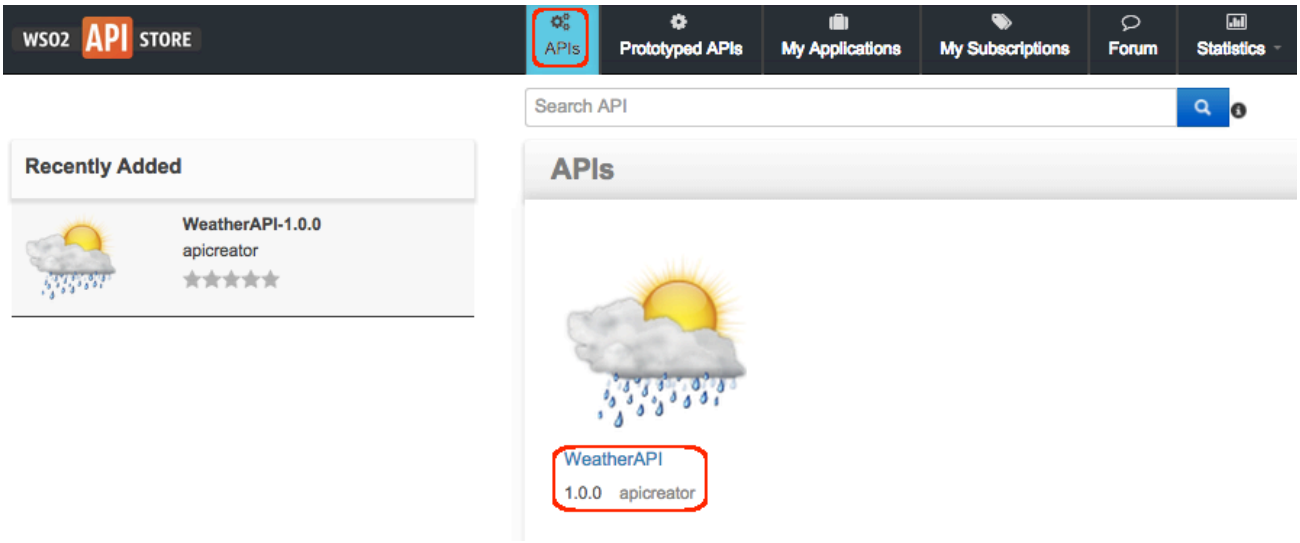


Let's publish this API.

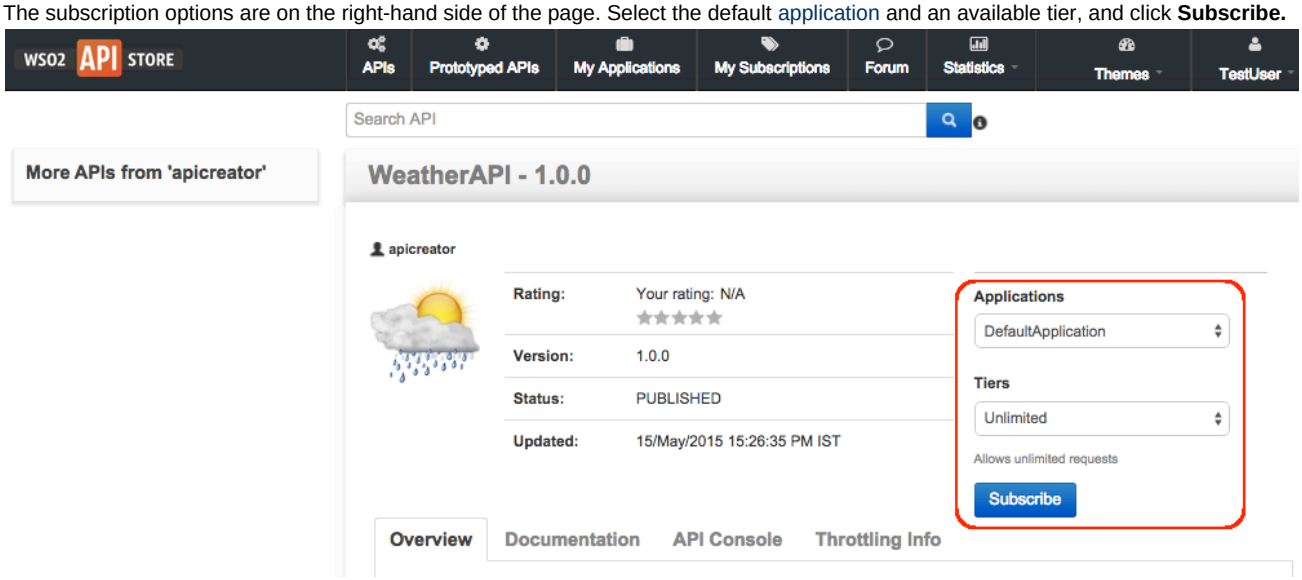
4. Go to the **Lifecycle** tab and note that the **State** is `PUBLISHED`. The API is already published to the API Store.



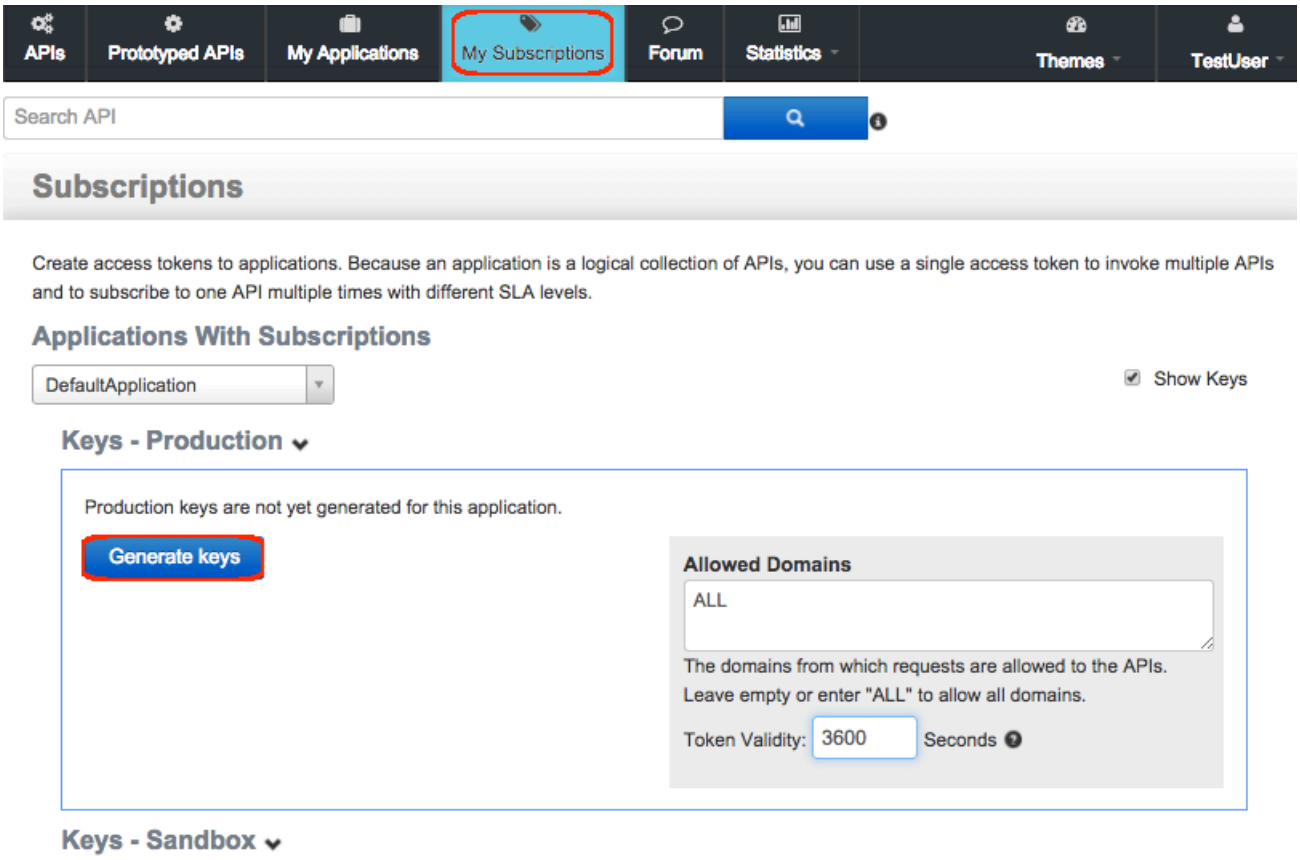
5. Log in to the API Store (`https://<hostname>:9443/store`) with **admin/admin** credentials and note that `WeatherAPI` is visible under the **APIs** menu. Click it to open the API.



6. The subscription options are on the right-hand side of the page. Select the default application and an available tier, and click **Subscribe.**

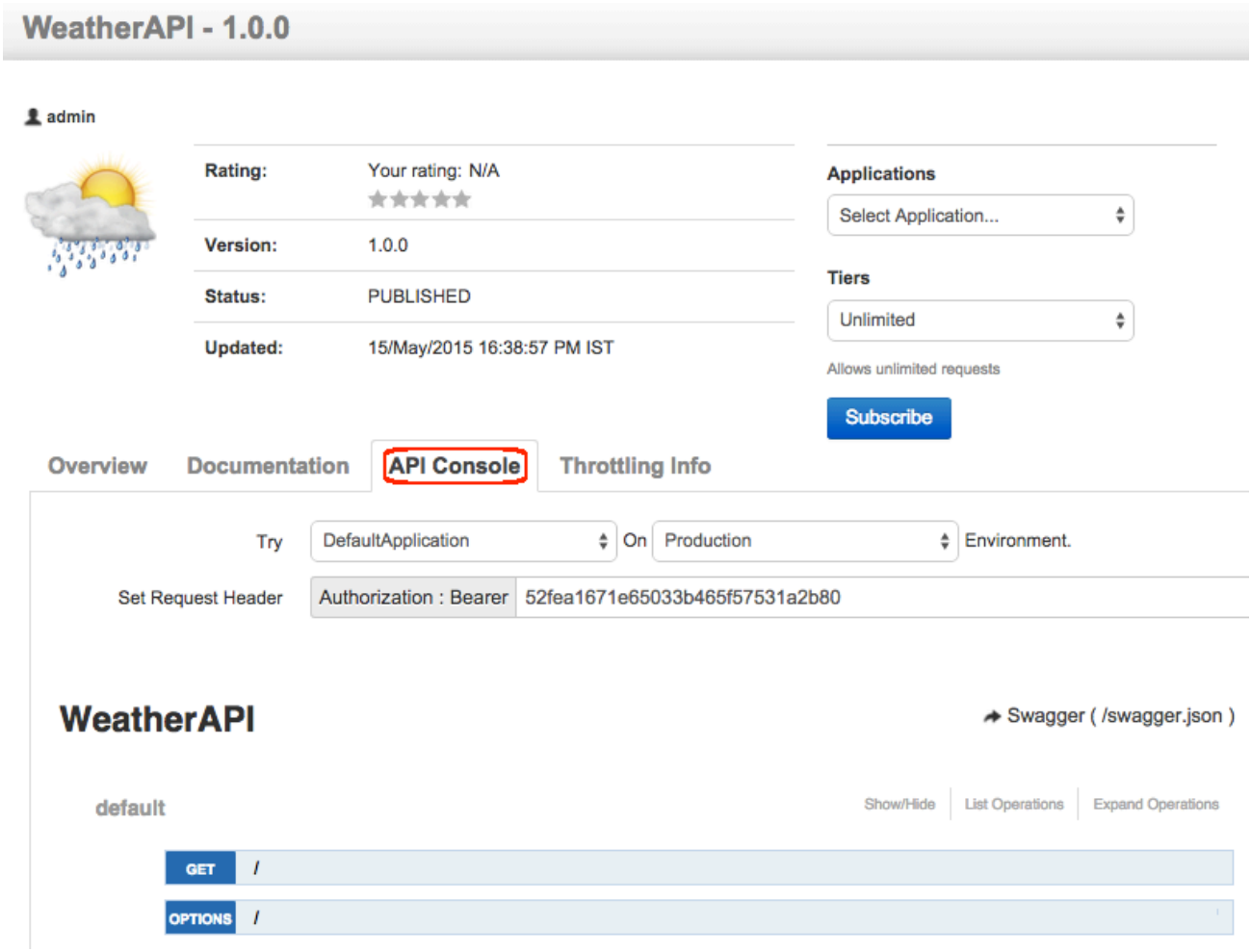


7. When the subscription is successful, choose to go to the **My Subscriptions** page and click the **Generate keys** button to generate an access token to invoke the API.

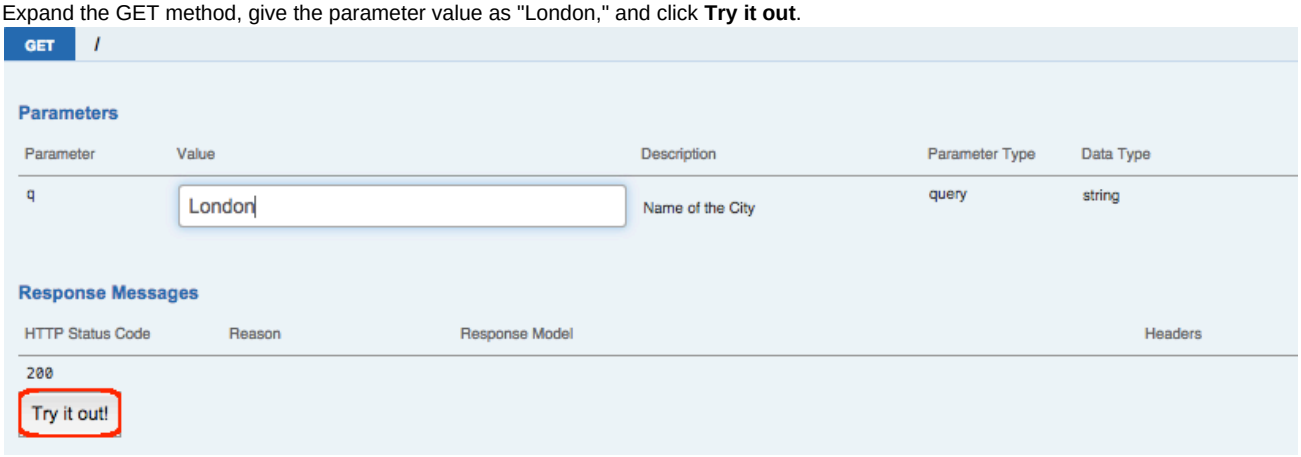


You are now successfully subscribed to an API. Let's invoke the API using the integrated Swagger-based API Console.

8. Click the **APIs** menu in the API Store again and then click the API to open it. When the API opens, click its **API Console** tab.



9. Expand the GET method, give the parameter value as "London," and click **Try it out**.



10. Note the response for the API invocation. It returns the weather in London.

You have deployed a sample API, published it to the API Store, subscribed to it, and invoked the API using our integrated API Console.

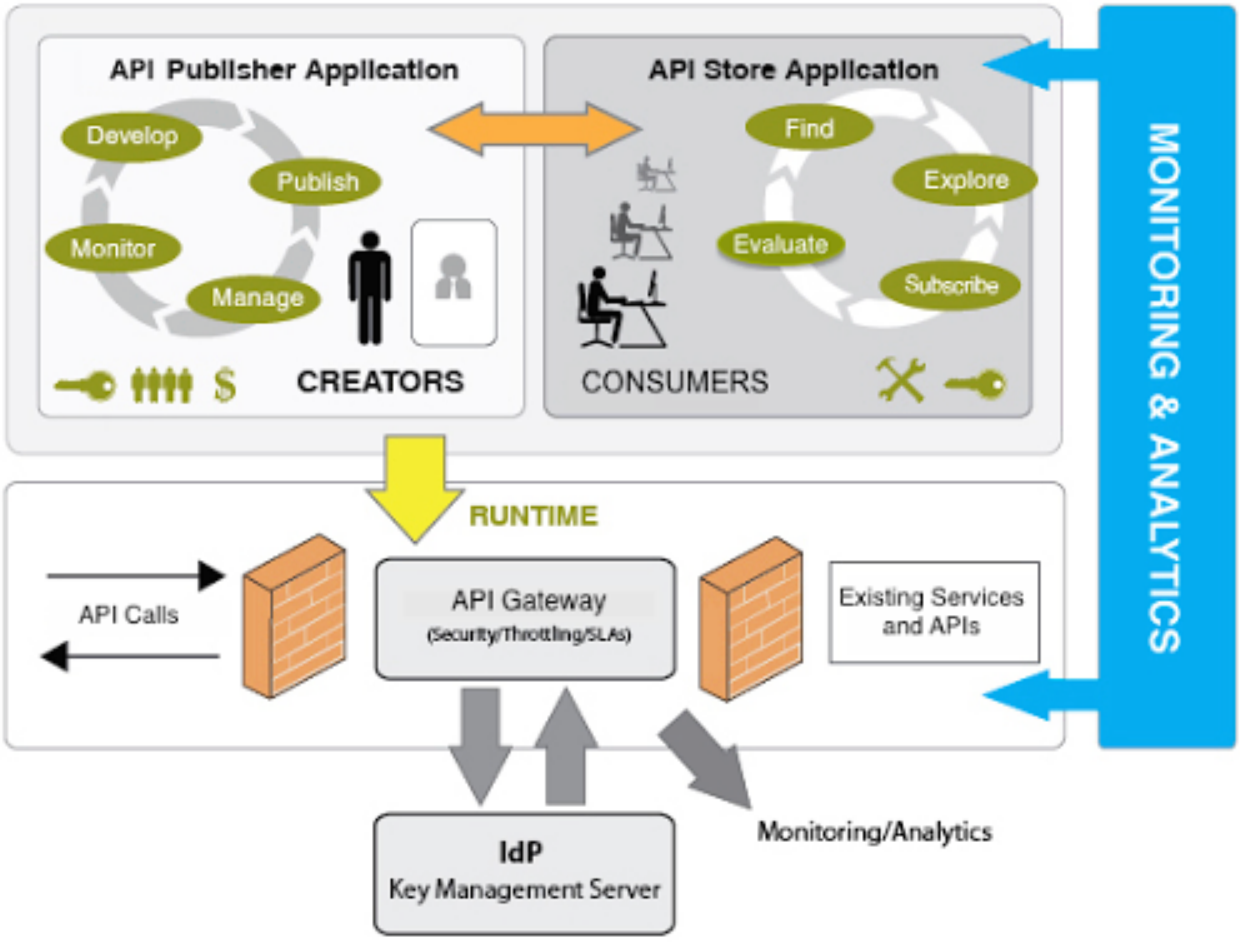
Understanding the API Manager concepts

Before we look into the API management activities in detail, let's take a look at the basic API management concepts.
[Components] [Users and roles] [API lifecycle] [Applications] [Throttling tiers] [API keys] [API resources]

Components

The API Manager comprises the following components:

- **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.
- **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**: 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.`
- Additionally, statistics are provided by the monitoring component, which integrates with WSO2 BAM.



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. The publisher is also interested in usage patterns for APIs and has access to all API statistics.
- **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 life cycles that are independent of the backend services they rely on. This life cycle is exposed in the API Publisher Web interface 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.

You can manage the API and service life cycles in the same governance registry/repository and automatically link them. This feature is available in WSO2 Governance Registry (version 4.5 onwards).

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. They define the throttling limits enforced by the API Gateway, e.g., 10 TPS (transactions per second). You define the list of tiers that are available for a given API at the publisher level. The API Manager comes with three predefined tiers (`Gold/Silver/Bronze`) and a special tier called `Unlimited`, which you can disable by editing the `<TierManagement>` eleme nt of the `<APIM_HOME>/repository/conf/api-manager.xml` file.

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 deployed on many different 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 NtBQkXoKElu0H1a1fQ0DWfo6IX4a,`") 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, you open the `<APIM_HOME>/repository/conf /identity.xml` file and change the value of the element `<ApplicationAccessTokenDefaultValidityPeriod>`. If you set a negative value, the token never expires.

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.

Application user access tokens have a fixed expiration time, which is 60 minutes by default. You can update it to a longer time by editing the `<App licationAccessTokenDefaultValidityPeriod>` element in the `<APIM_HOME>/repository/conf/identity.xml` file.

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, OPTIONS, DELETE. You can give multiple values at once.
- **uri-template**: A URI template as defined in http://tools.ietf.org/html/rfc6570. E.g., /phoneverify/<phoneNumber>.
- **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, or 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.

Deep diving into the API Manager

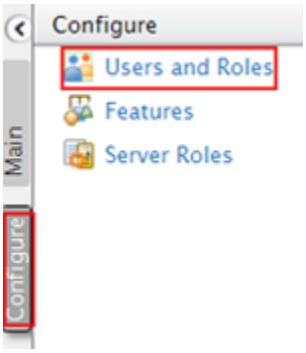
Let's take a look at the typical API management activities in detail:

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

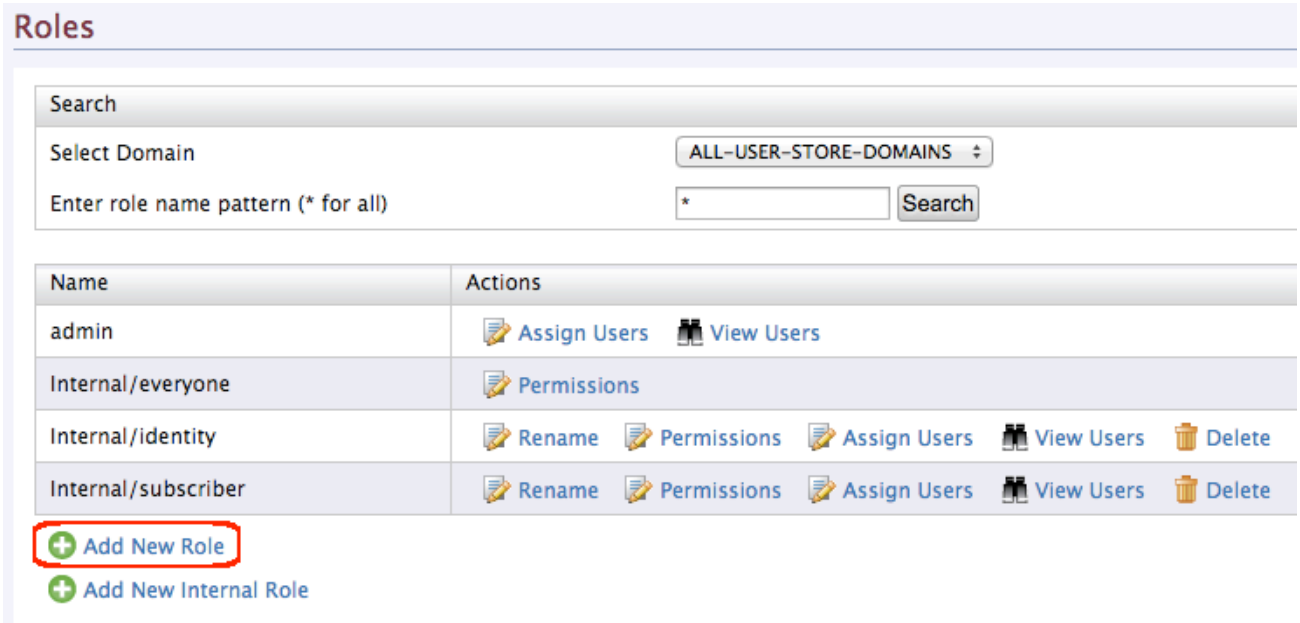
Creating users and roles

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

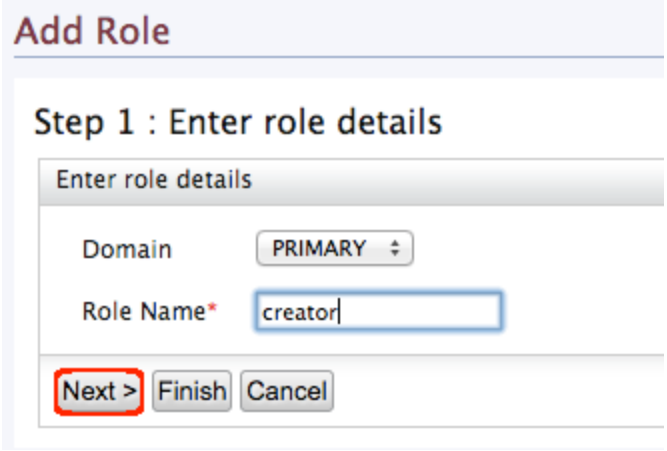
1. Log in to the Management Console (`https://<hostname>:9443/carbon`) of the API Manager using admin/admin credentials.
2. Select the **Users and Roles** menu under the **Configure** menu.



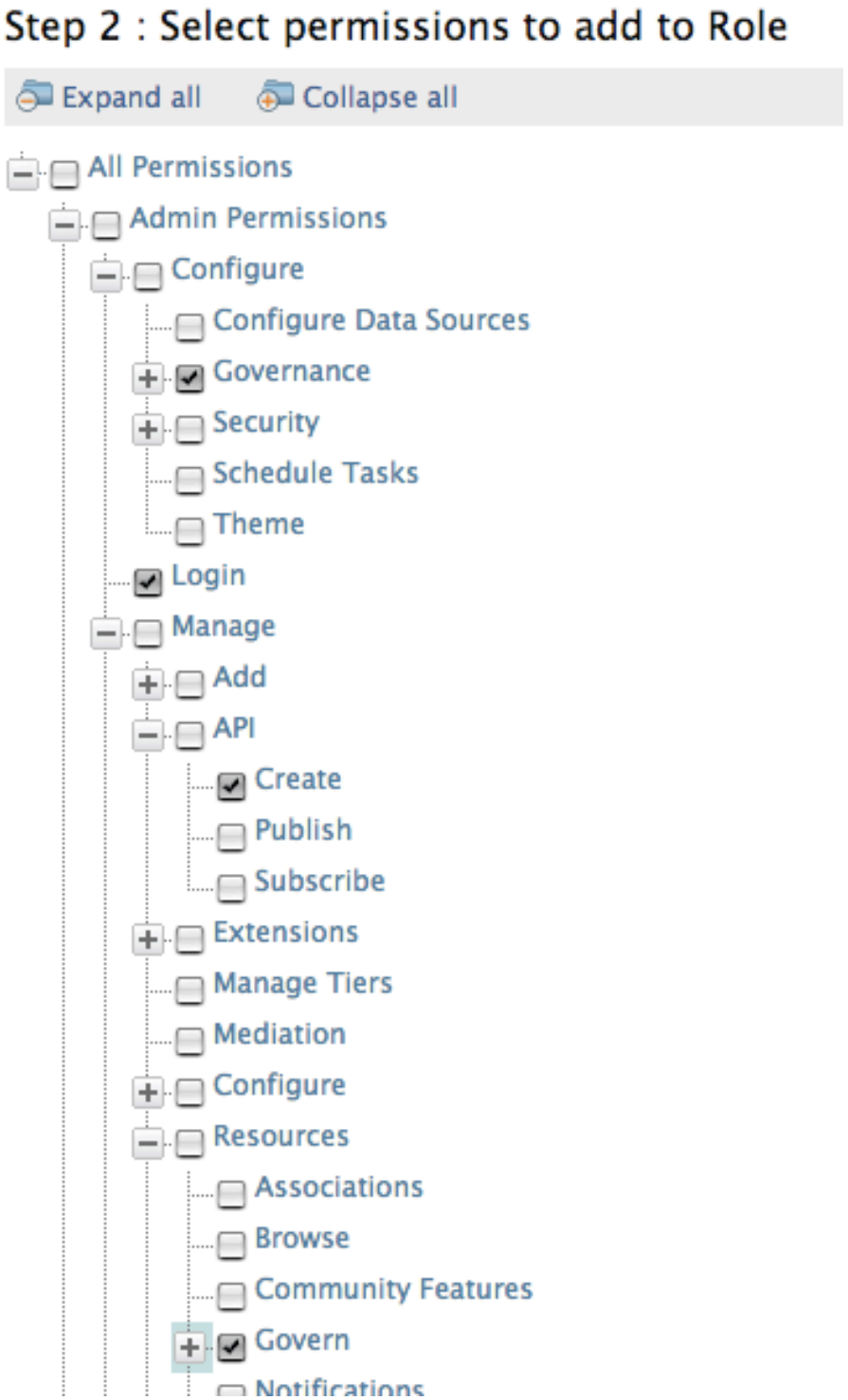
3. Click the **Roles** link and then click **Add New Role**.



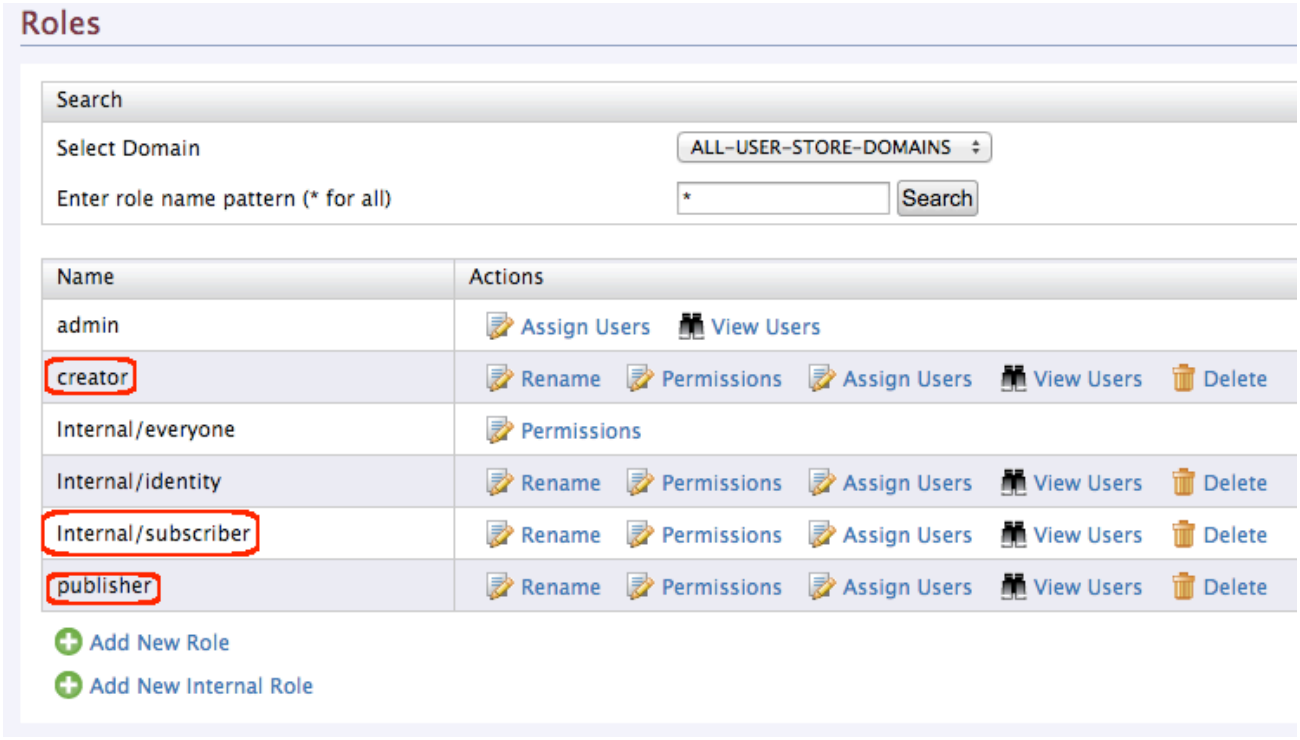
4. Give the role name as `creator` and click **Next**.



5. A list of permissions opens. Select the following and click **Finish**.
    - Configure > Governance and all underlying permissions
    - Login
    - Manage > API > Create
    - Manage > Resources > Govern and all underlying permissions

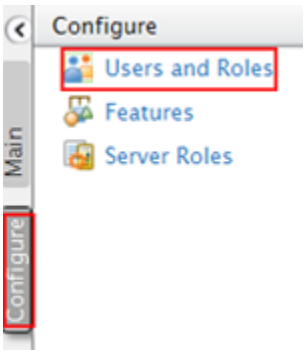


6. Similarly, create the `publisher` role with the following permissions.
    - Login
    - Manage > API > Publish
7. Note that the API Manager comes with the `subscriber` role available by default. It has the following permissions:
    - Login
    - Manage > API > Subscribe
8. Note that the roles you added (creator, internal/subscriber, and publisher) are now displayed under **Roles**.

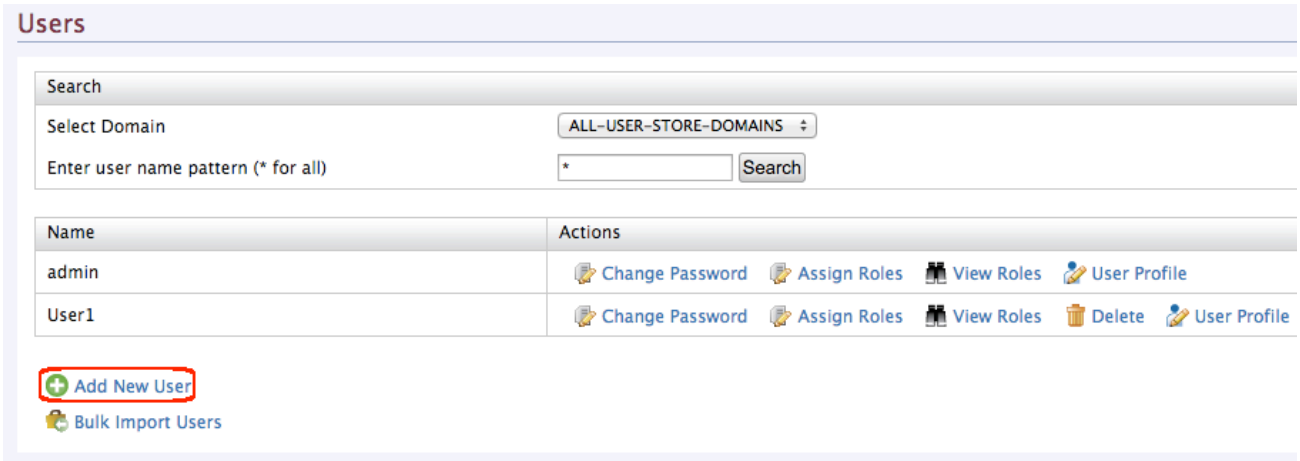


Let's create users for each of the roles.

9. Click the **Users and Roles** menu under the **Configure** menu again.

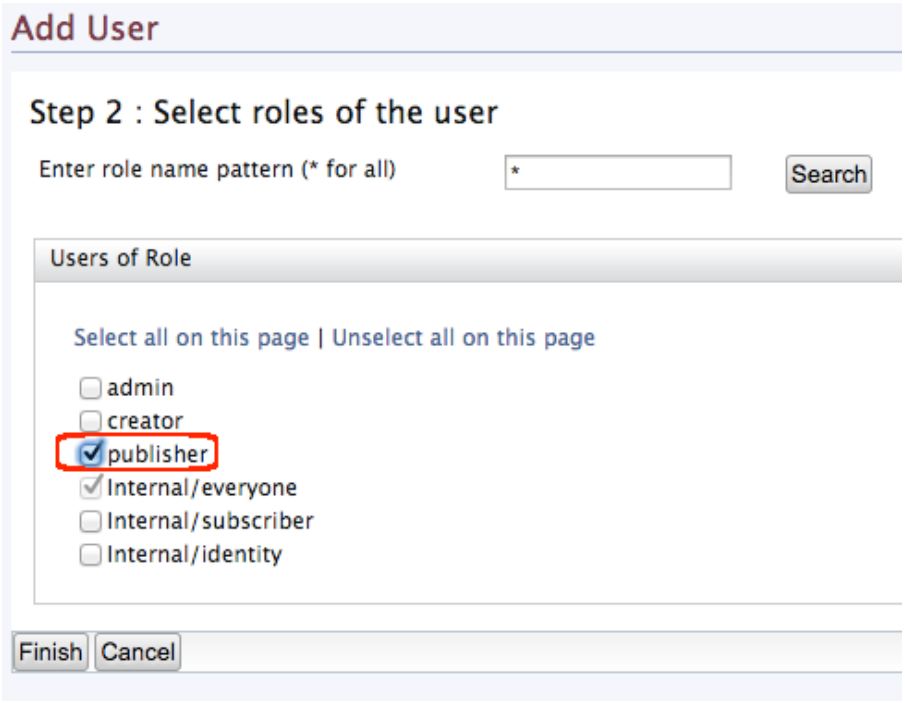


10. Click the **Users** link and then click **Add New User.**



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**.

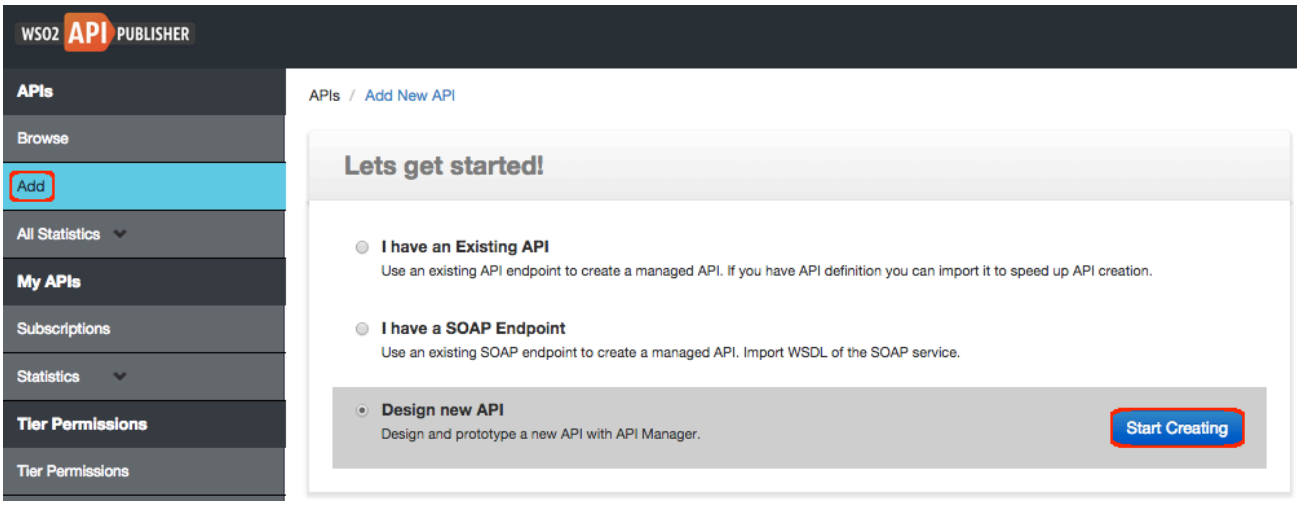


13. Similarly, create a new user by the name `apicreator` and assign the creator role.

Creating an API from scratch

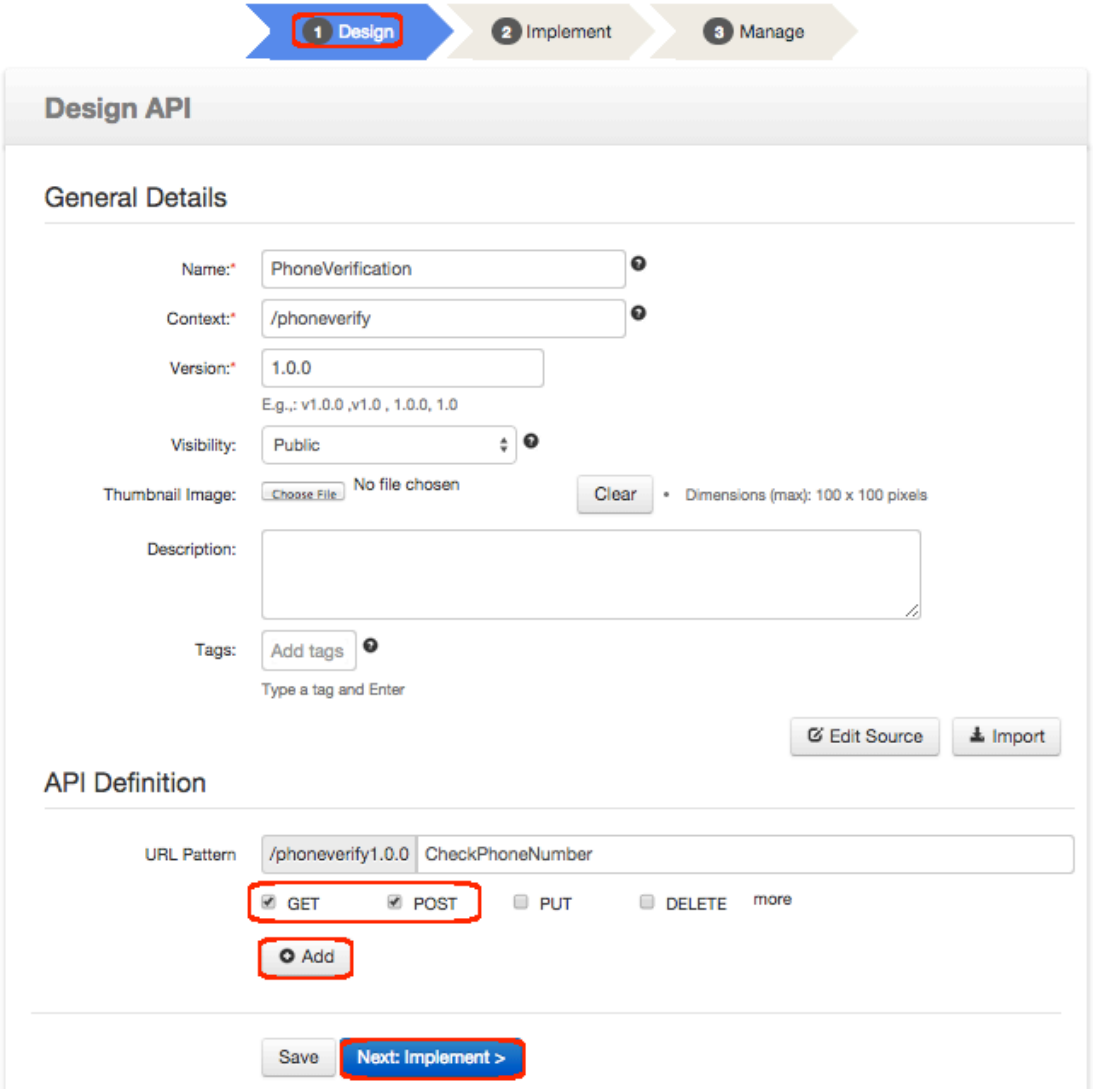
Let's create an API from scratch.

1. Log in to the API Publisher (`https://<hostname>:9443/publisher`) as **`apicreator`**.
2. Select the option to design a new API and click **Start Creating**.



3. Give the information in the table below and click **Implement** to move on to the next page.

Field	Sample value
Name	PhoneVerification
Context	/phoneverify
Version	1.0.0
Visibility	Public
API Definition	• URL pattern: CheckPhoneNumber • Request types: GET, POST

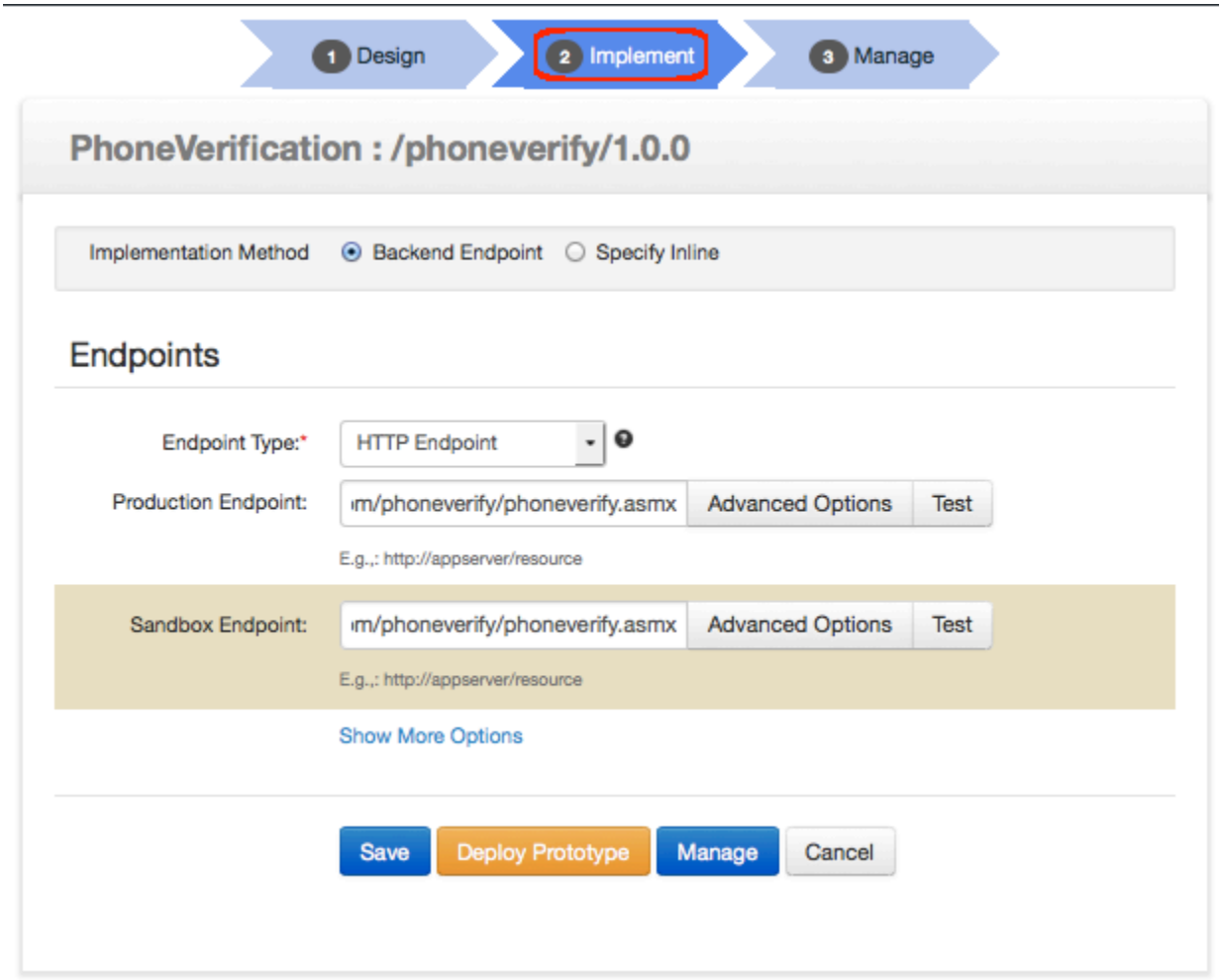


4. Select the **Managed API** option.

5. Give the following information in the **Implement** tab that opens and click **Manage** once you are done.

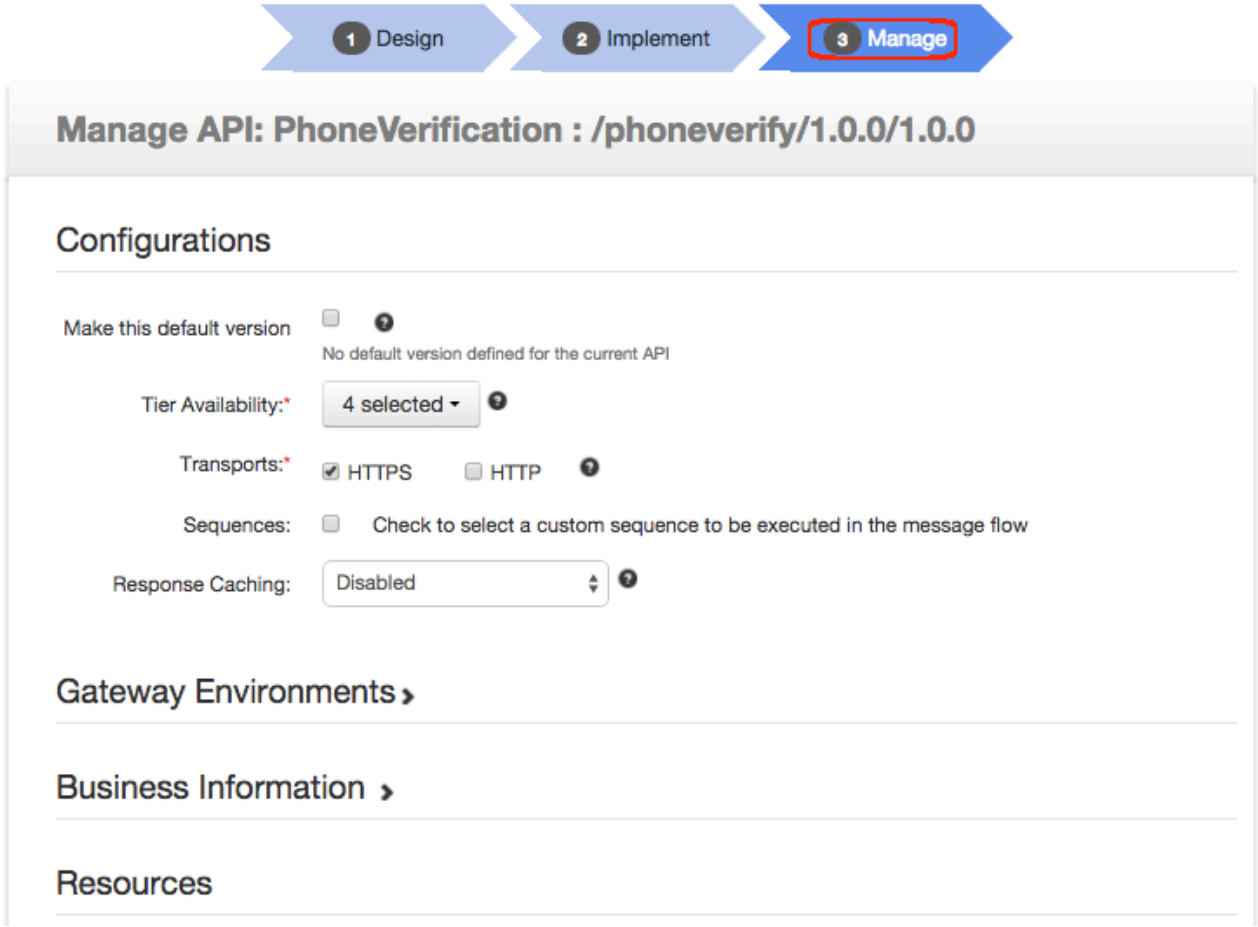
Field	Sample value

Endpoint type	HTTP
Production endpoint	In this guide, we work with a service exposed by the Cdyne services provider. We use their phone validation service, which has SOAP and REST interfaces. Endpoint is http://ws.cdyne.com/phoneverify/phoneverify.asmx. This sample service has two operations: `CheckPhoneNumber` and `CheckPhoneNumbers`. Let's use `CheckPhoneNumber` here.



6. Click **Manage** to go to the `Manage` tab and provide the following information. Leave default values for the rest of the parameters in the UI.

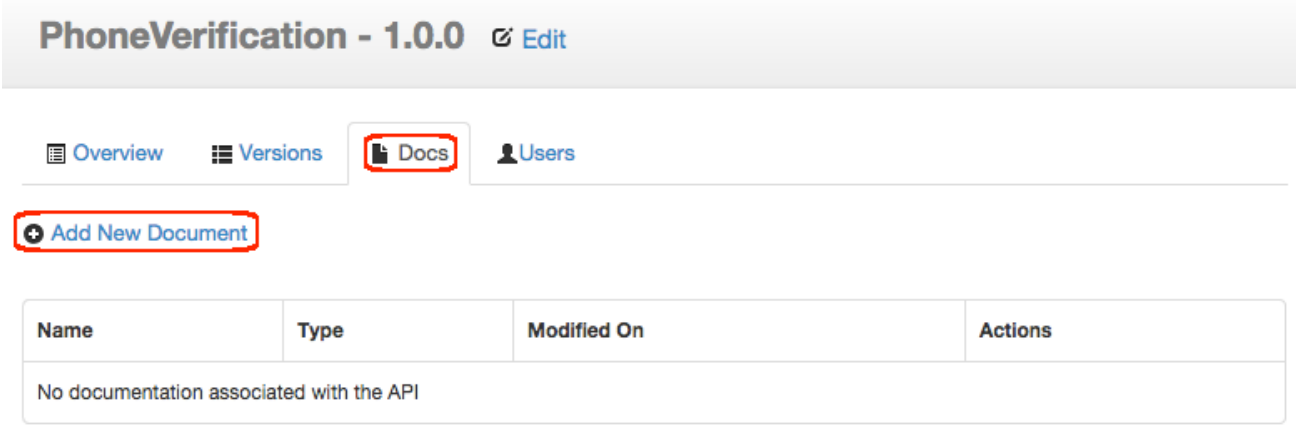
Field	Value	Description
Tier Availability	<Select all available tiers>	The API can be available at different levels of service. They allow you to limit the number of successful hits to an API during a given period of time.



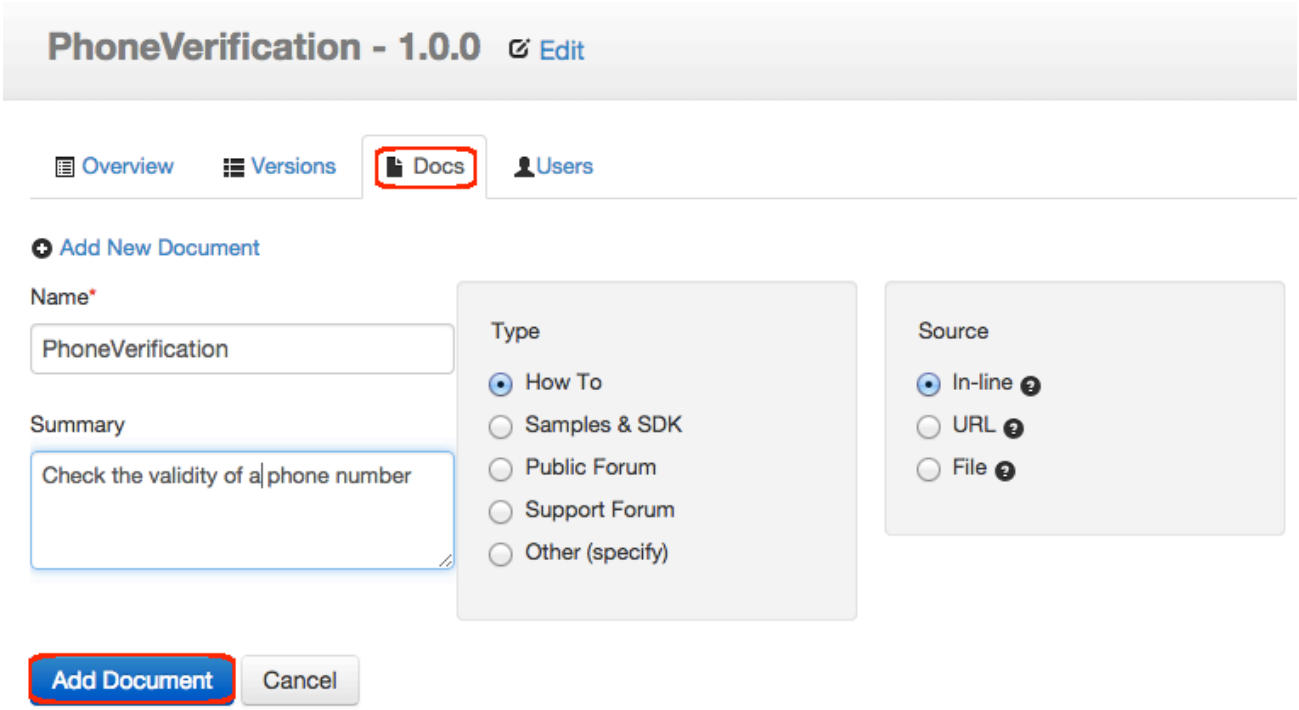
7. Once you are done, click **Save**.

Adding API documentation

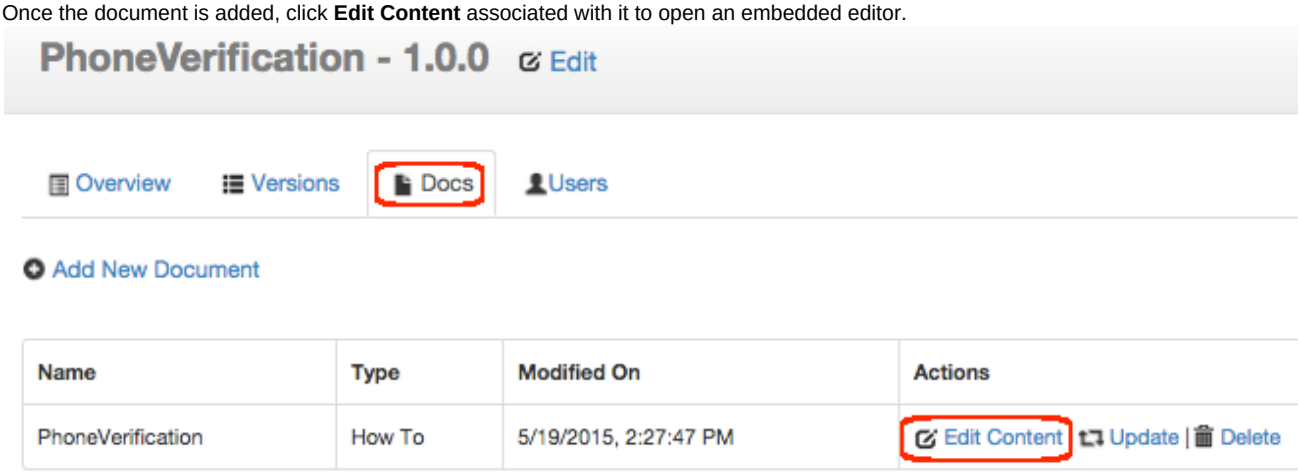
1. After saving the API, click its thumbnail in the API Publisher to open it.
2. Click on the API's **Docs** tab and click the **Add New Document** link.



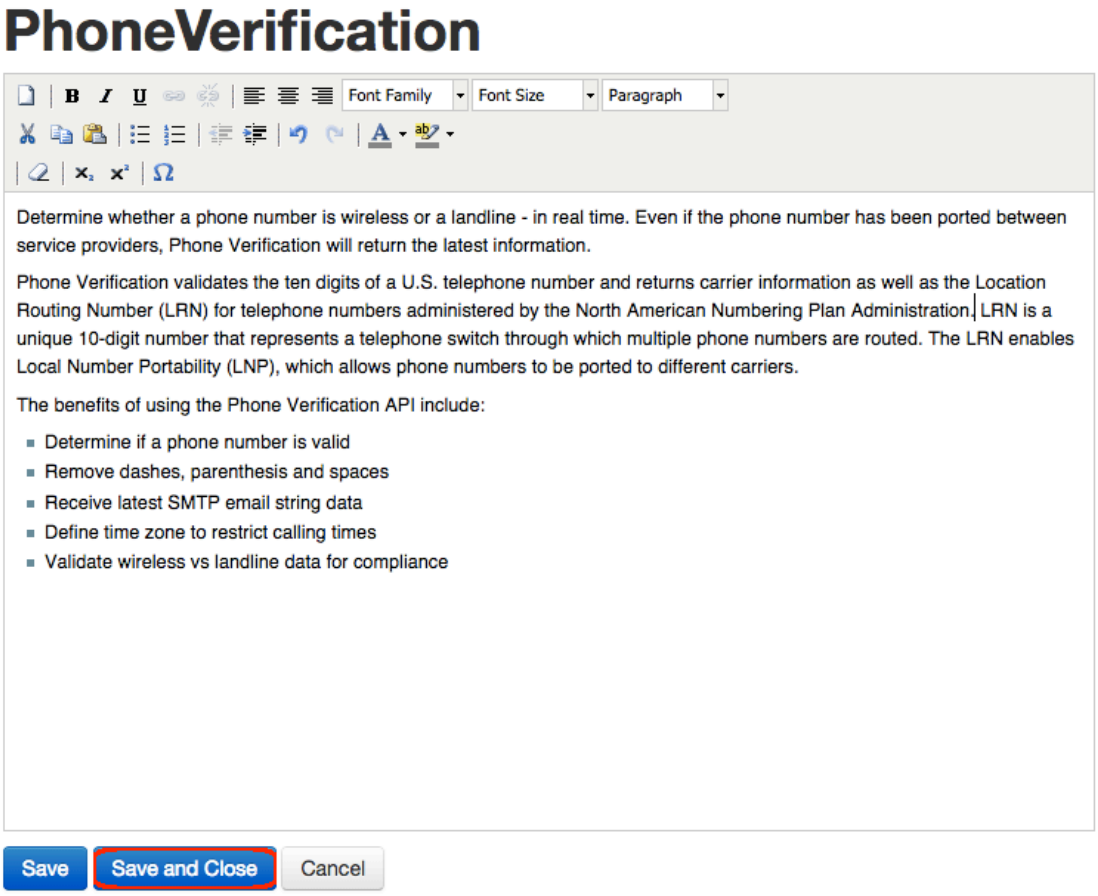
3. The document options appear. Note that you can create documentation inline, via a URL, or as a file. For inline documentation, you can edit the content directly from the API publisher interface. You get several documents types:
    - How To
    - Samples and SDK
    - Public forum / Support forum (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**.



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



6. Enter your API's documentation.



Adding interactive documentation

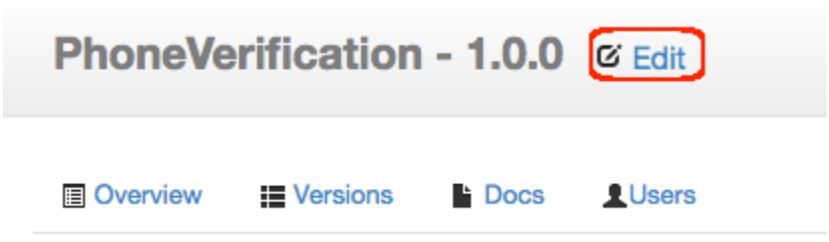
WSO2 API Manager has an integrated Swagger UI, which is part of the Swagger project.

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 a services in the same way that interfaces describe lower-level programming code.

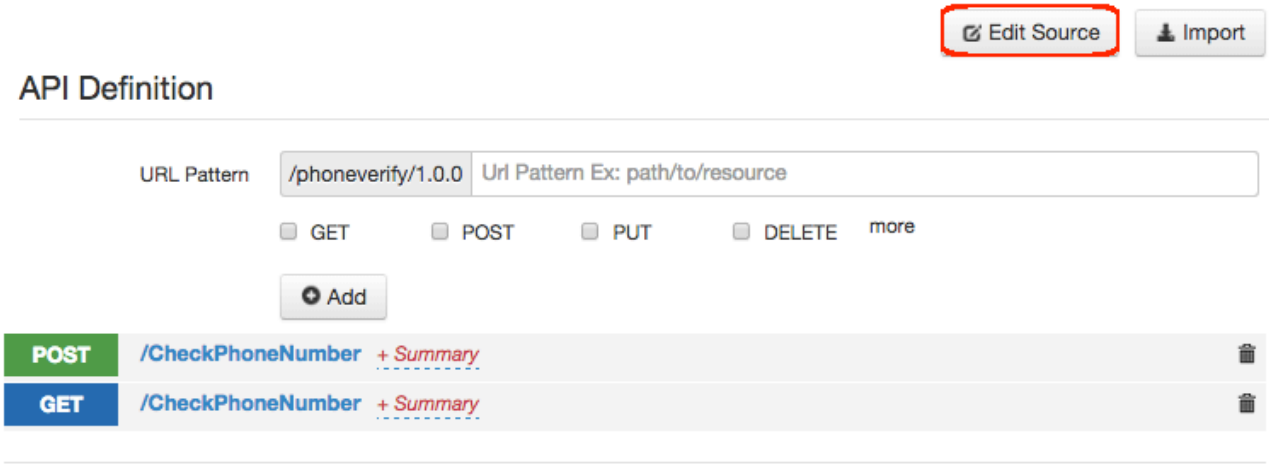
The Swagger UI is a dependency-free collection of HTML, JavaScript, and CSS that dynamically generates documentation from a Swagger-compliant API. Swagger-compliant APIs give you interactive documentation, client SDK generation, and more discoverability. The Swagger UI has JSON code, and its UI facilitates 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.

Also, see the Swagger 2.0 specification.

1. Open the API Publisher (`https://<hostname>:9443/publisher`) and log in as **`apicreator`**.
2. Click the `PhoneVerification` API to open it, and then click the **Edit** link right next to the API's name. This opens the API in its edit mode.

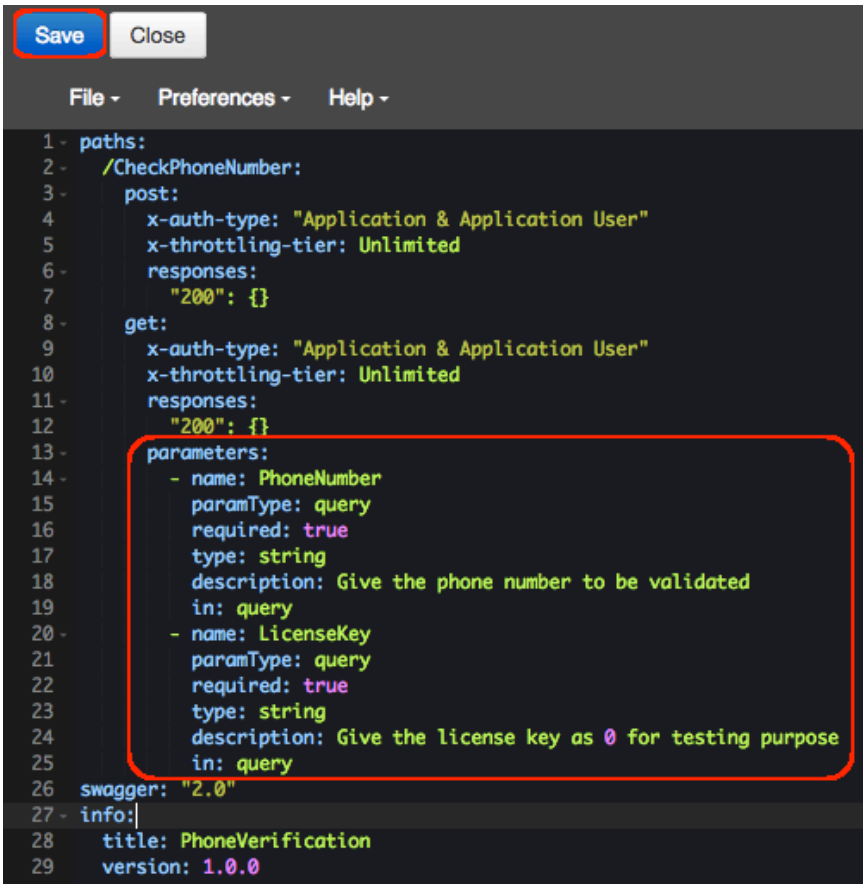


3. Click the **Edit Source** button near the **Resources** section.

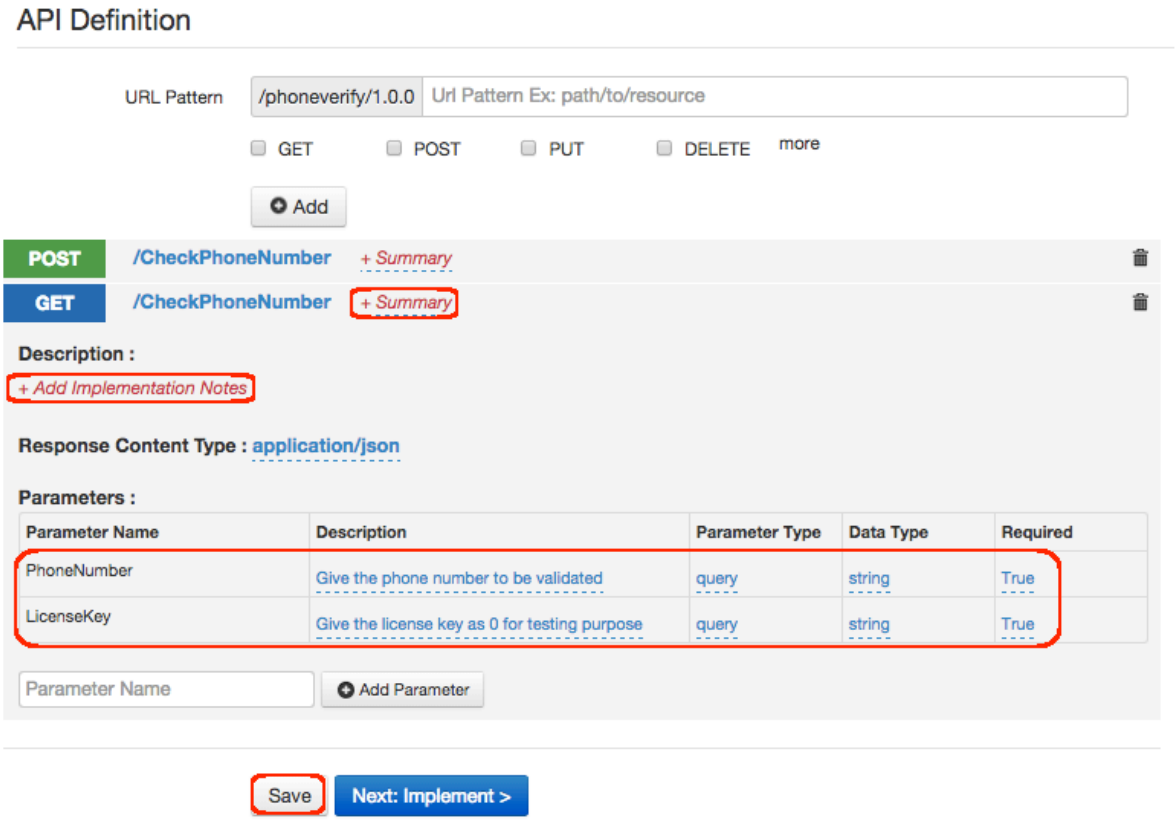


4. The JSON code of the API opens in a separate page. Expand its GET method, add the following parameters and click **Save**.

```
parameters:
        - name: PhoneNumber
          paramType: query
          required: true
          type: string
          description: Give the phone number to be validated
          in: query
        - name: LicenseKey
          paramType: query
          required: true
          type: string
          description: Give the license key as 0 for testing purpose
          in: query
```



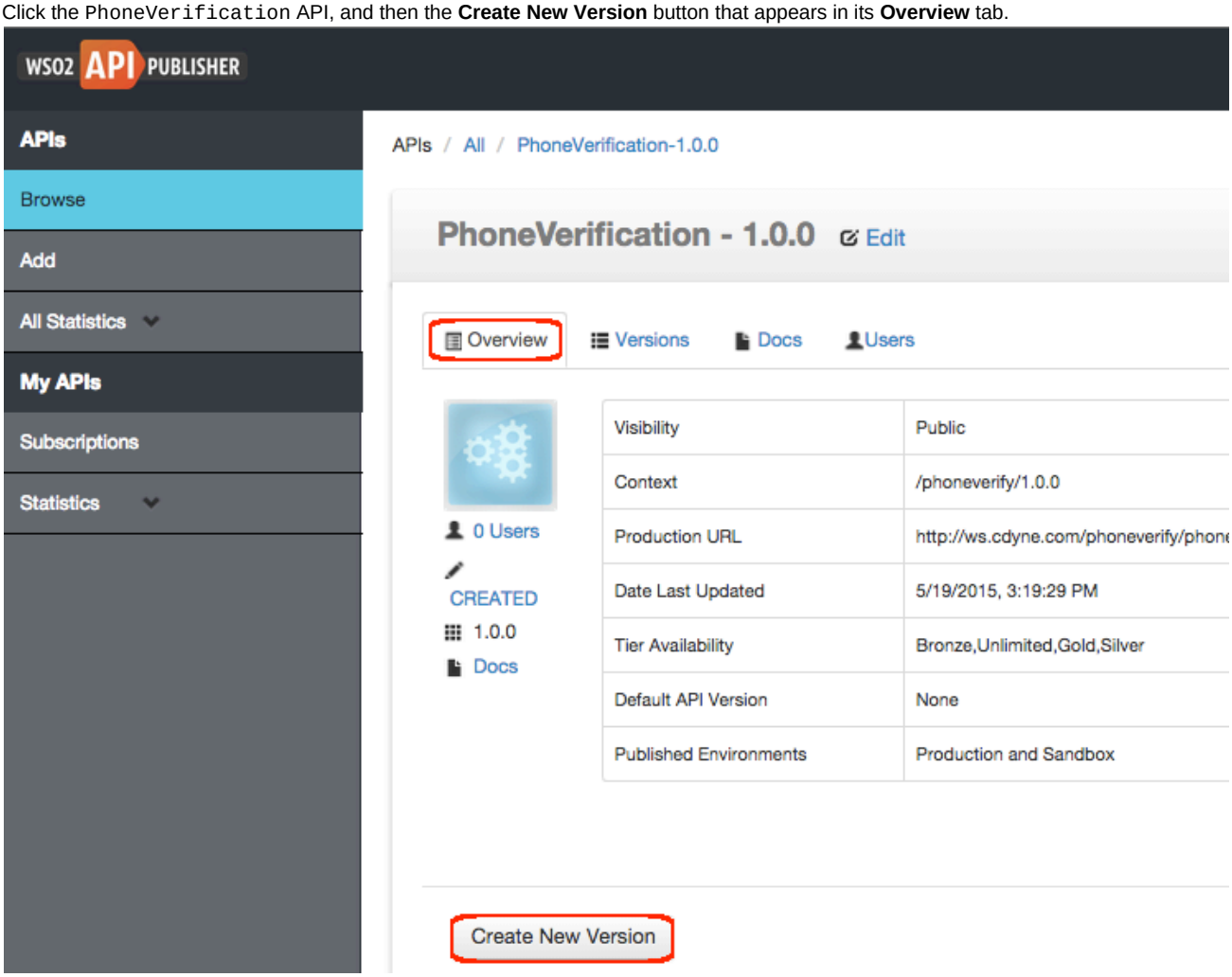
5. Back in the API Publisher, note that the changes you did appear in the API Console's UI. You can add more parameters and edit the summary/descriptions using the API Publisher UI as well. Once done, click **Save**.



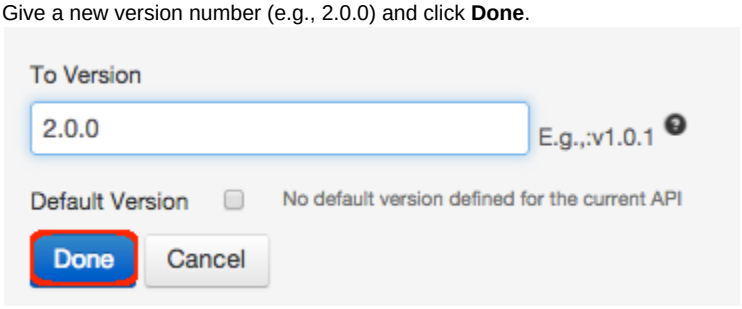
Versioning the API

Let's create a new version of this API.

1. Log in to the API Publisher as **`apicreator`** if you are not logged in already.
2. Click the `PhoneVerification` API, and then the **Create New Version** button that appears in its **Overview** tab.



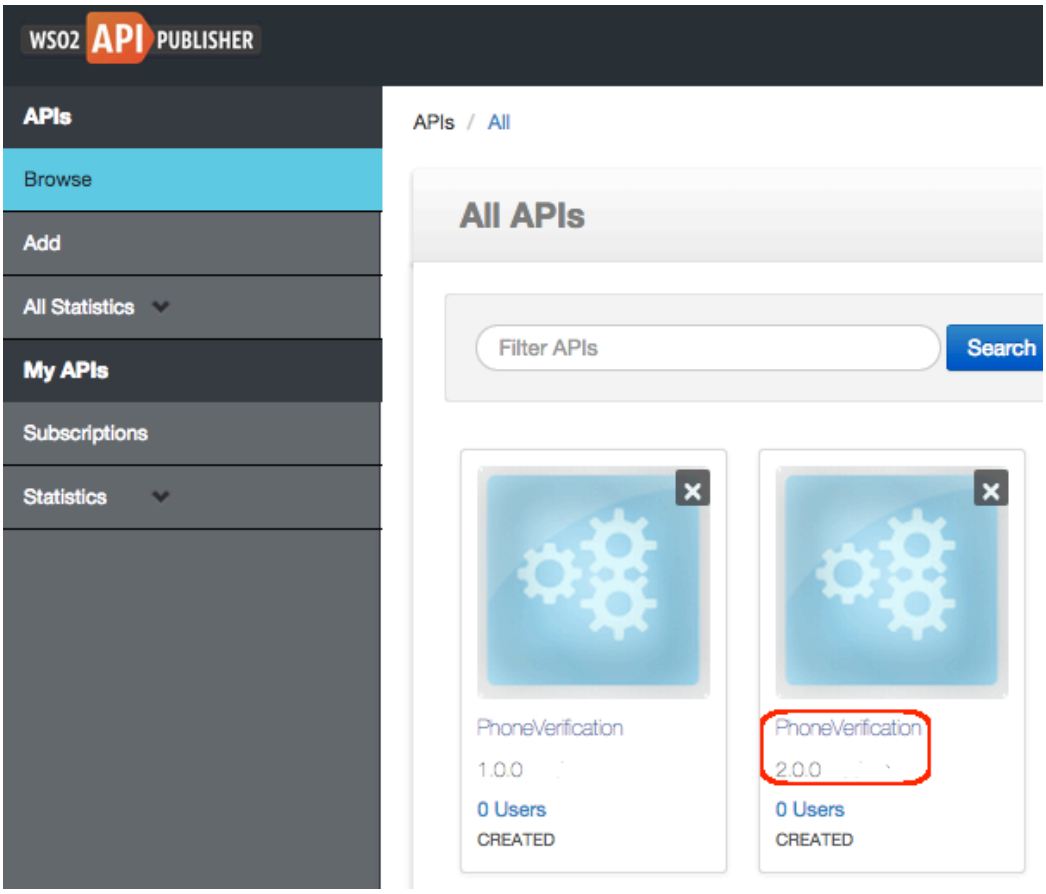
3. Give a new version number (e.g., 2.0.0) and click **Done**.



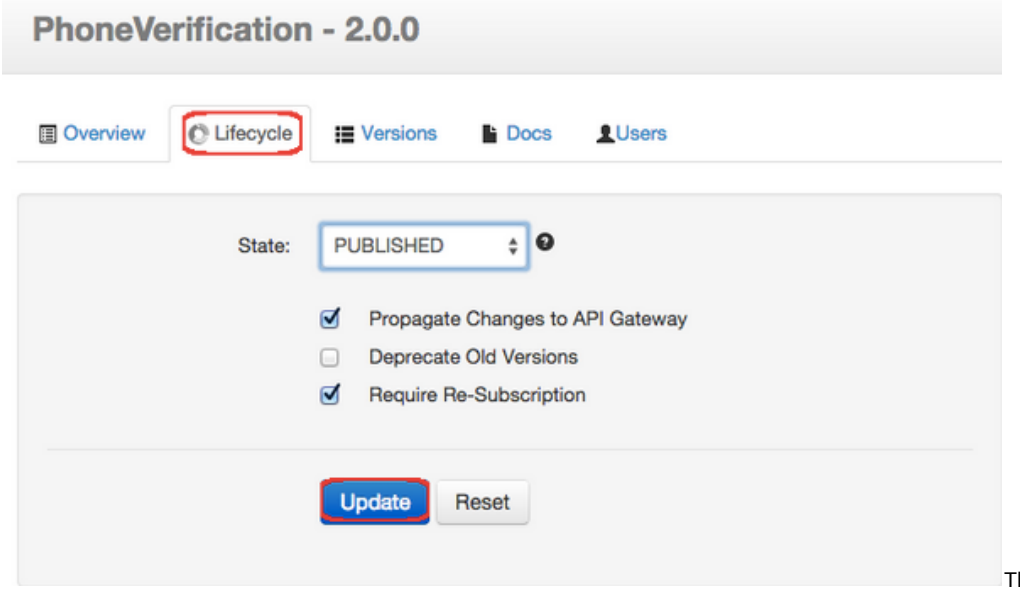
4. Note that the new version of the API is created in the API Publisher.

Publishing the API

1. Log 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.



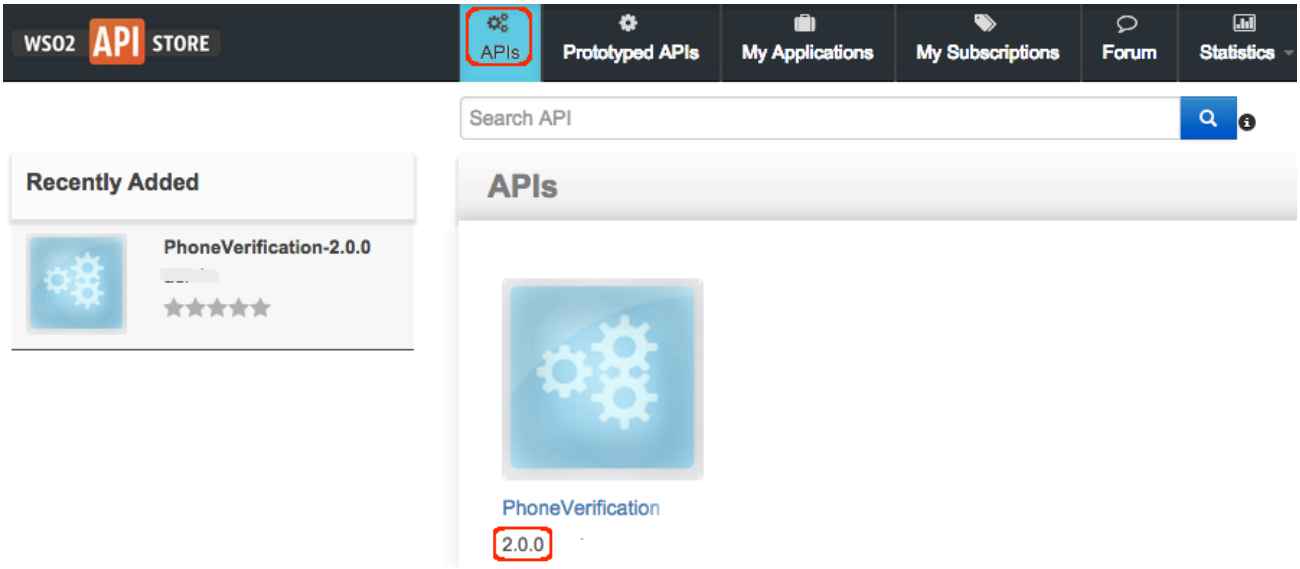
2. The API opens. Go to its **Lifecycle** tab, select the state as `PUBLISHED` from the drop-down list, and click **Update**.



The three check boxes mean the following:

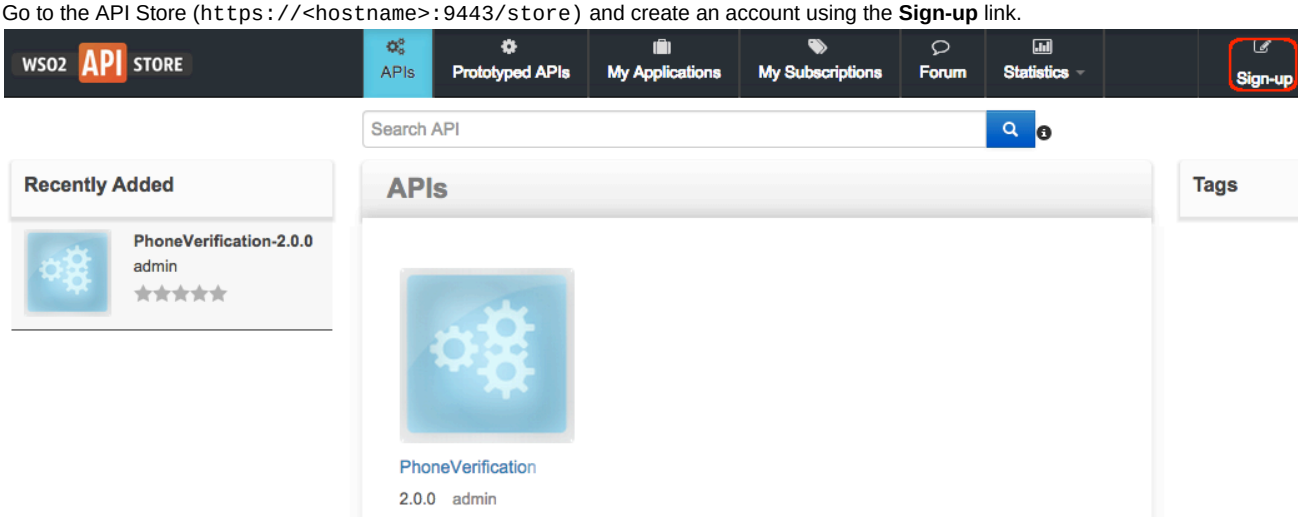
- **Propagate Changes to API Gateway**: Used to define an API proxy in the API Gateway runtime component, allowing the API to be exposed to the consumers via the API Gateway. If this option is left unselected, the API metadata will not change, and you will have to manually configure the API Gateway according to the information published in the API Store.
- **Deprecate Old Versions**: If selected, any prior versions of the API that are published will be set to the DEPRECATED state automatically.
- **Require Re-Subscription**: Invalidates current user subscriptions, forcing users to subscribe again.

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.

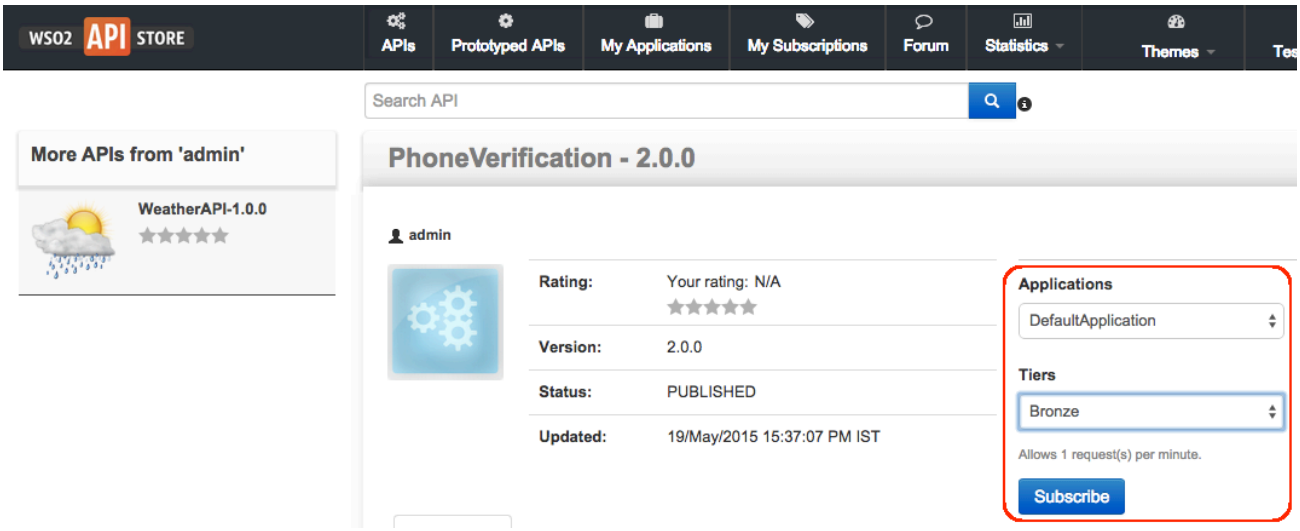


Subscribing to the API

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

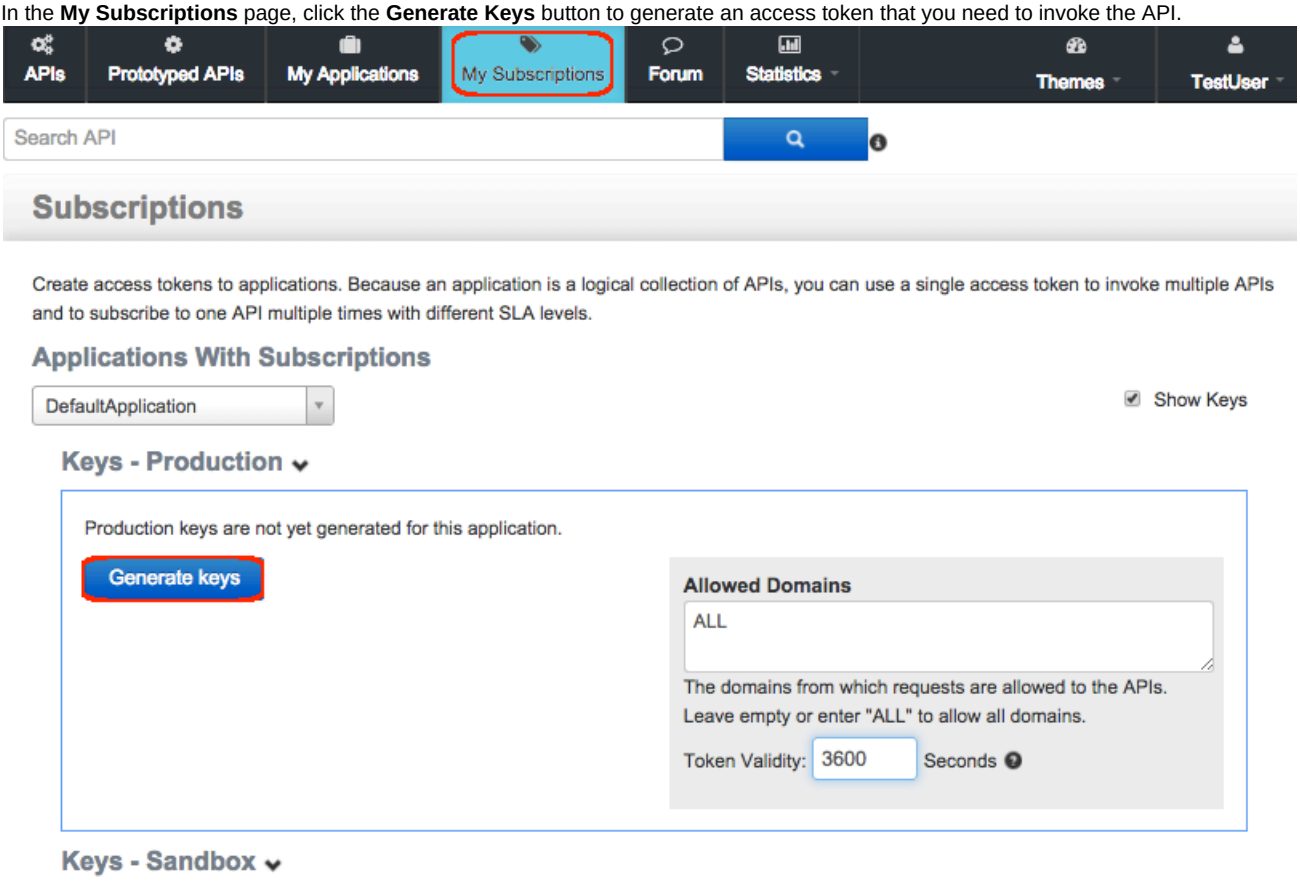


2. After signing up, log in to the API Store and click the `PhoneVerification 2.0.0` API that you published earlier.
3. Note that you can now see the subscription options on the right-hand side of the UI. Select the default application, select the `Bronze` tier and click **Subscribe**.



4. Once the subscription is successful, choose to go to the **My Subscriptions** page.

5. In the **My Subscriptions** page, click the **Generate Keys** button to generate an access token that you need to invoke the API.

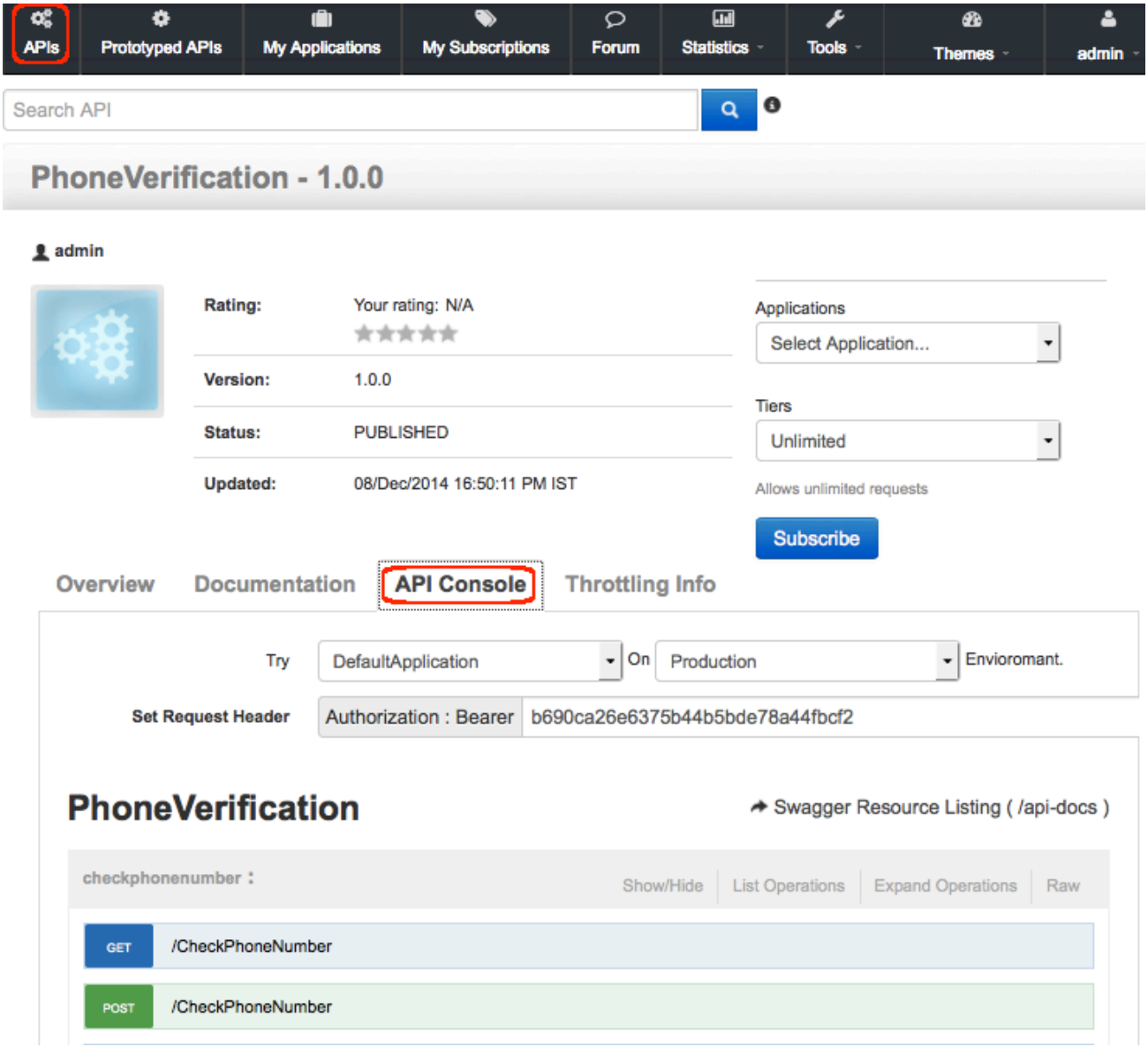


> **Tip** : You can set a token validity period in the given text box. By default, it is set to one hour. If you set a minus value (e.g., -1), the token will never expire.

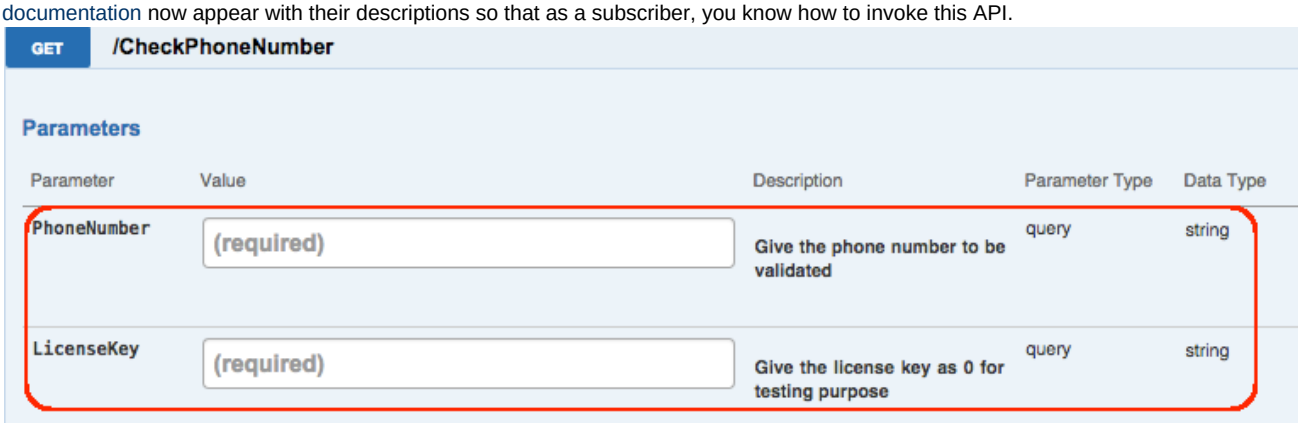
You are now successfully subscribed to an API. Let's invoke it.

Invoking the API

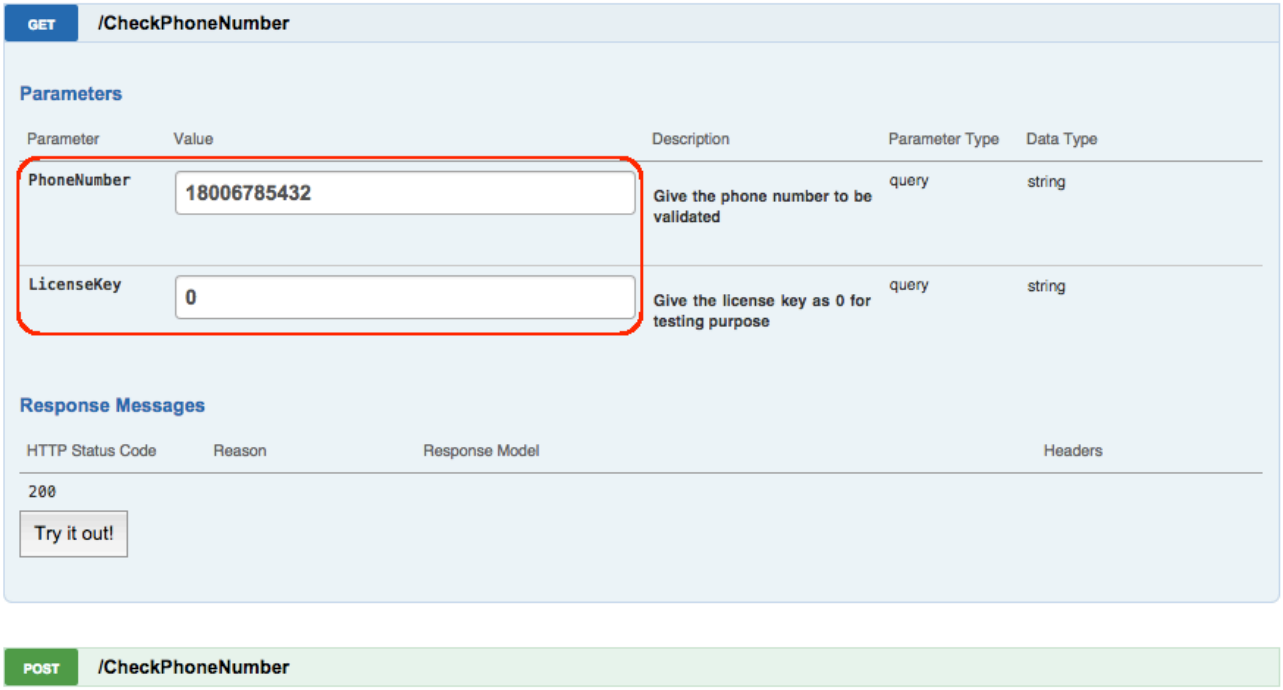
1. Click the **APIs** menu in the API Store and then click on the API that you want to invoke. When the API opens, go to its **API Console** tab.



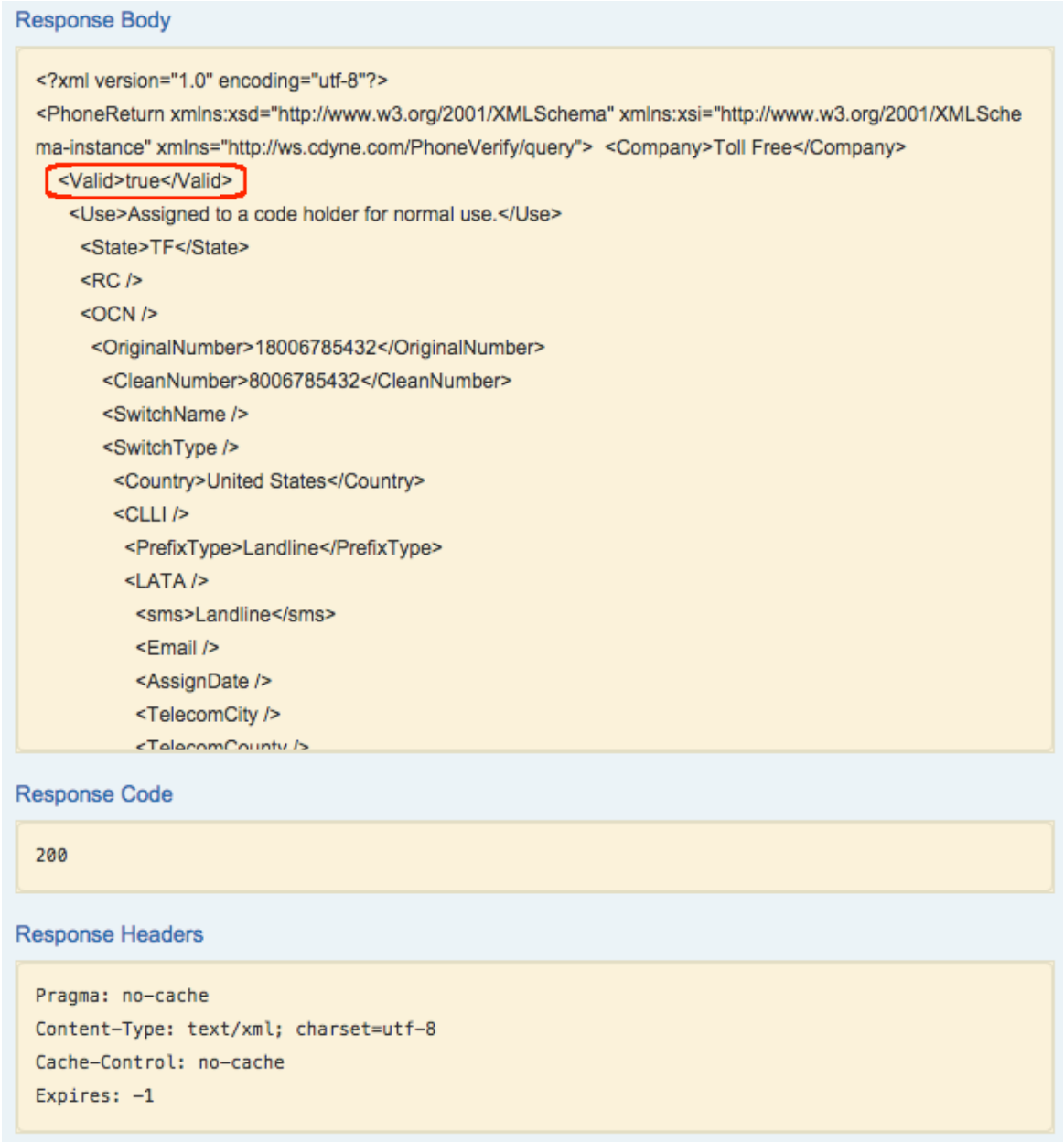
2. Expand the GET method of the resource `CheckPhoneNumber`. Note the parameters that you added when creating the interactive documentation now appear with their descriptions so that as a subscriber, you know how to invoke this API.



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



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



You have invoked an API using the API Console.

Monitoring APIs and viewing statistics

Both the API publisher and store provide several statistical dashboards. Some of them are as follows:

- Number of subscriptions per API (across all versions of an API)
- Number of API calls being made per API (across all versions of an API)
- The subscribers who did the last 10 API invocations and the APIs/versions they invoked
- Usage of an API and from which resource path (per API version)
- Number of times a user has accessed an API
- The number of API invocations that failed to reach the endpoint per API per user
- API usage per application
- Users who make the most API invocations, per application
- API usage from resource path, per application

The steps below explain how to configure WSO2 Business Activity Monitor (BAM) 2.5.0 with the API Manager. The statistics in these dashboards are based on data from the BAM. The steps below explain how to configure WSO2 BAM 2.5.0 with the API Manager.

If you are on **Windows**, note the following:

- If you installed the JDK in Program Files in the Windows environment, avoid the space by using PROGRA~1 when specifying

environment variables for JAVA_HOME and PATH. Otherwise, the server throws an exception.
- Install Cygwin (http://www.cygwin.com). WSO2 BAM depends on Apache Hadoop, which requires Cygwin in order to run on Windows. Install at least the basic net (OpenSSH,tcp_wrapper packages) and security-related Cygwin packages. After Cygwin installation, update the PATH variable with `C:/cygwin/bin`. If you already have WSO2 BAM running, you must restart it now.

Let's do the configurations first.

1. Apply an offset of 3 to the default BAM port by editing the `<BAM_HOME>/repository/conf/carbon.xml` file. This makes the BAM server run on port 9446 instead of the default port 9443, and avoids port conflicts when multiple WSO2 products run on the same host.

```
<Offset>3</Offset>
```

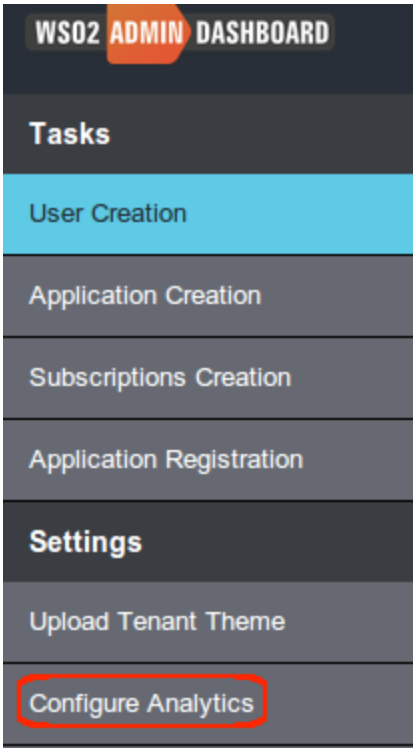
2. Download MySQL from https://www.mysql.com/ and install it in your server.
3. Go to the command-line and issue the following commands to connect to the MySQL server and create a database (e.g., TestStatsDB). This database is used to save the statistical data collected by the BAM. You do not need to create any tables in it.

```
mysql -u <username> -p <password> -h <host_name or IP>;
CREATE DATABASE TestStatsDB;
```

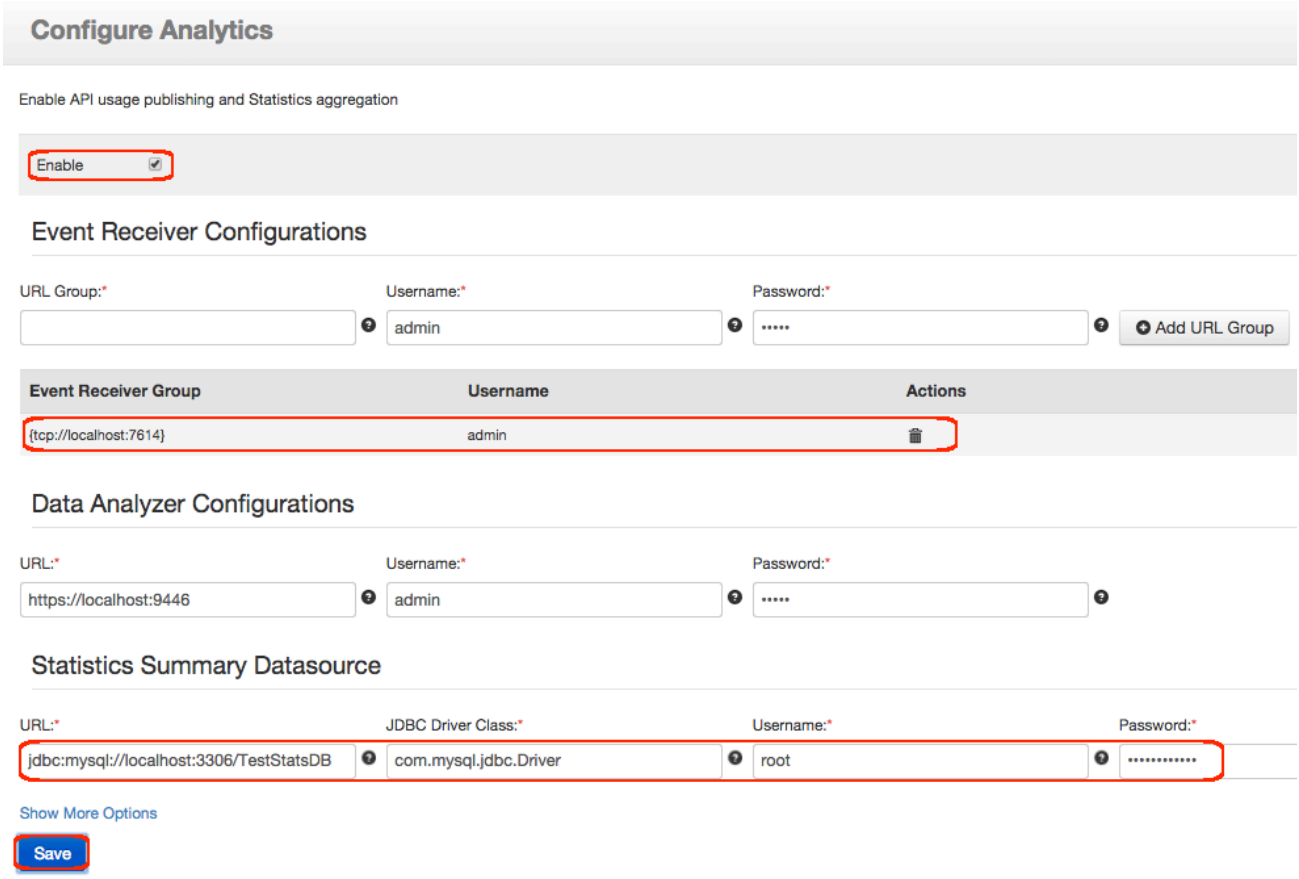
4. Save the MySQL connector JAR inside both the `<APIM_HOME>/repository/components/lib` and `<BAM_HOME>/repository/components/lib` folders.
5. Give the datasource definition under the `<datasource>` element in the `<BAM_HOME>/repository/conf/datasources/master-datasources.xml` file. For example,

```
<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:mysql://localhost:3306/TestStatsDB</url>
           <username>db_username</username>
           <password>db_password</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>
```

6. Start the BAM server by running either of the following commands in the command line:
   - On Windows: `<PRODUCT_HOME>\bin\wso2server.bat --run`
   - On Linux/Solaris/Mac OS: `sh <PRODUCT_HOME>/bin/wso2server.sh`
7. Start the API Manager and log in to its Admin Dashboard Web application (`https://<Server Host>:9443/admin-dashboard`) with **admin/admin** credentials.
8. Click the **Configure Analytics** menu.

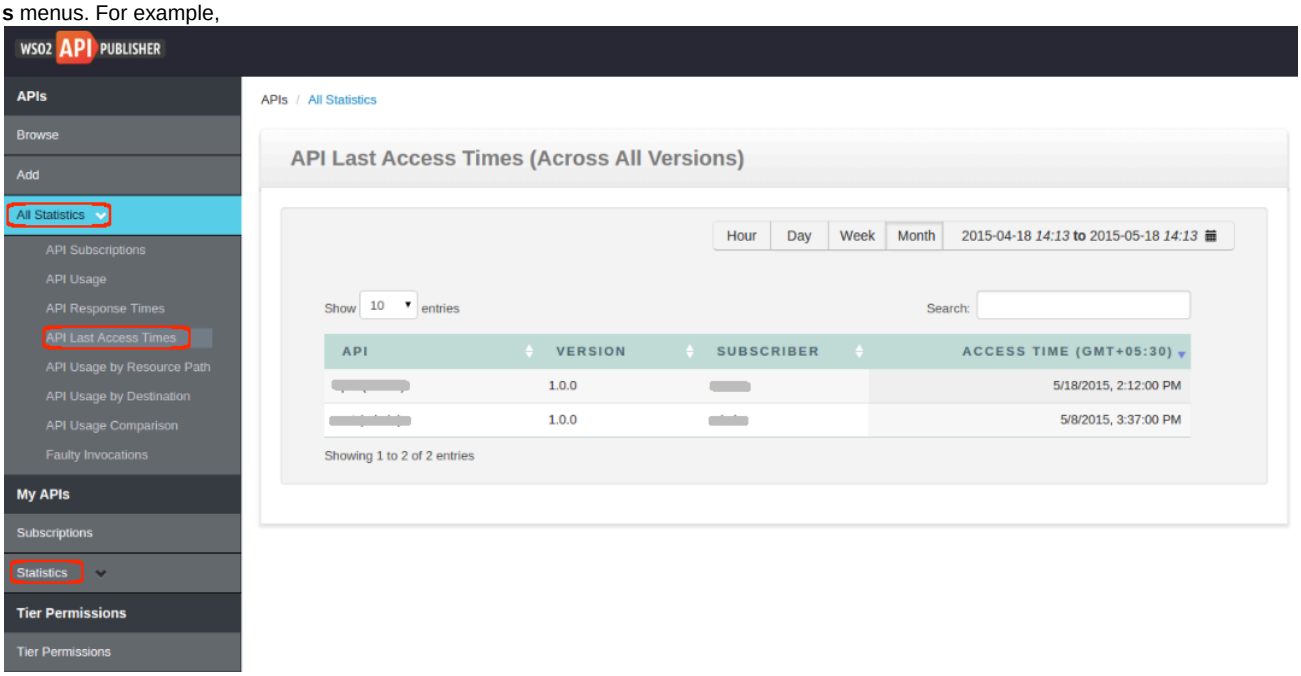


9. Select the **Enable** check box to enable statistical data publishing and add the following:
    - Add a URL group as tcp://<BAM server IP>:7614 and click **Add URL Group**.
    - Fill the details under **Statistics Summary Database** according to the information you added to the `master-datasources.xml` file in step 4.



10. Click **Save**. The BAM deploys the Analytics toolbox, which describes the information collected, how to analyze the data, and the location of the database where the analyzed data is stored.
11. Invoke several APIs to generate some statistical data and wait a few seconds.

12. Connect to the API Publisher as a creator or publisher and click the statistical dashboards available under the **All Statistics** and **Statistics** menus. For example,



The **All Statistics** menu is available for both API creators and publishers. It shows statistics of all APIs. The **Statistics** menu is available for API creators to see statistics of only the APIs created by them.

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, please see the User Guide and the Admin Guide of the API Manager documentation.