Exercise 12

API Management and Governance including Analytics

Prior Knowledge
RESTful services
Previous ESB exercises

Objectives
Understand API management and key issuing. Understand Business Activity Monitoring. Be able to configure the API Manager and Business Activity Monitor, and use OAuth2 Bearer Tokens

Software Requirements
OpenJDK 1.8
WSO2 API Manager 1.10.0 (WSO2 AM)
WSO2 Data Analytics Server 3.0.1 (WSO2 DAS)
Node.js and npm (and other existing APIs)

1) We are going to use the payment API that we wrote earlier. In addition, we have a simple sandbox test service that is written in node.js.

2) To make this easier, I have created a simple docker composition that includes the ESB mediation, the Tomcat Payment WAR and a sandbox service implemented in node.js. This command will execute the whole lot using docker-compose:

   curl –L http://freo.me/dc-serv | sh

3) Start the WSO2 BAM Server:
   a. `cd ~/servers/wso2bam-2.4.1/`
   b. `bin/wso2server.sh`

4) Start the WSO2 API Manager:
   a. `cd ~/servers/wso2am-1.7.0`
   b. `bin/wso2server.sh`

5) Wait until both are started and then check that you can access the admin screens:
   a. https://localhost:9447/carbon (AM)
   b. https://localhost:9448 (BAM)
      Because we have not got a "real" TLS/SSL certificate, your browser will complain about these websites. You will need to persuade your browser to move on! (Browser dependent).

6) To create the users and roles in the API Manager, you log in to the management console as an administration user (default credentials: **admin/admin**).

   To speed things up we have already created the following users and roles.

Username	Role	Password
charlie	creator	password
peter	publisher	password

To see what we did go look at the Appendix A.

7) An API creator uses the API provider Web application to create and publish APIs into the API Store. In this section, we explain how to create an API and attach documentation to it.

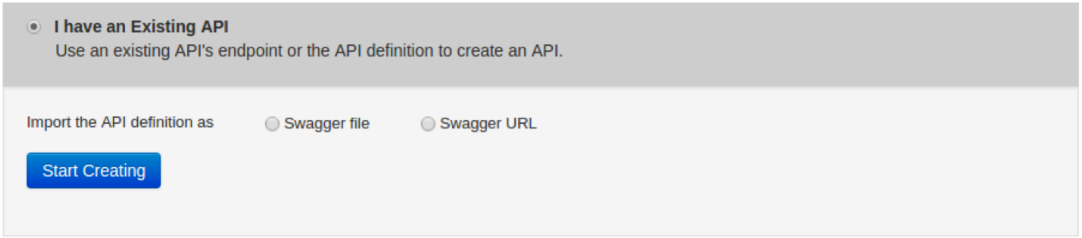
   In this guide, we work with a service exposed by the node.js server running on port 8001 on the VM. Let's create this API and add it to the API Store.

   Open the API Publisher system (https://localhost:9447/publisher) and log in as **charlie/password**

8) Click the **New API** button.

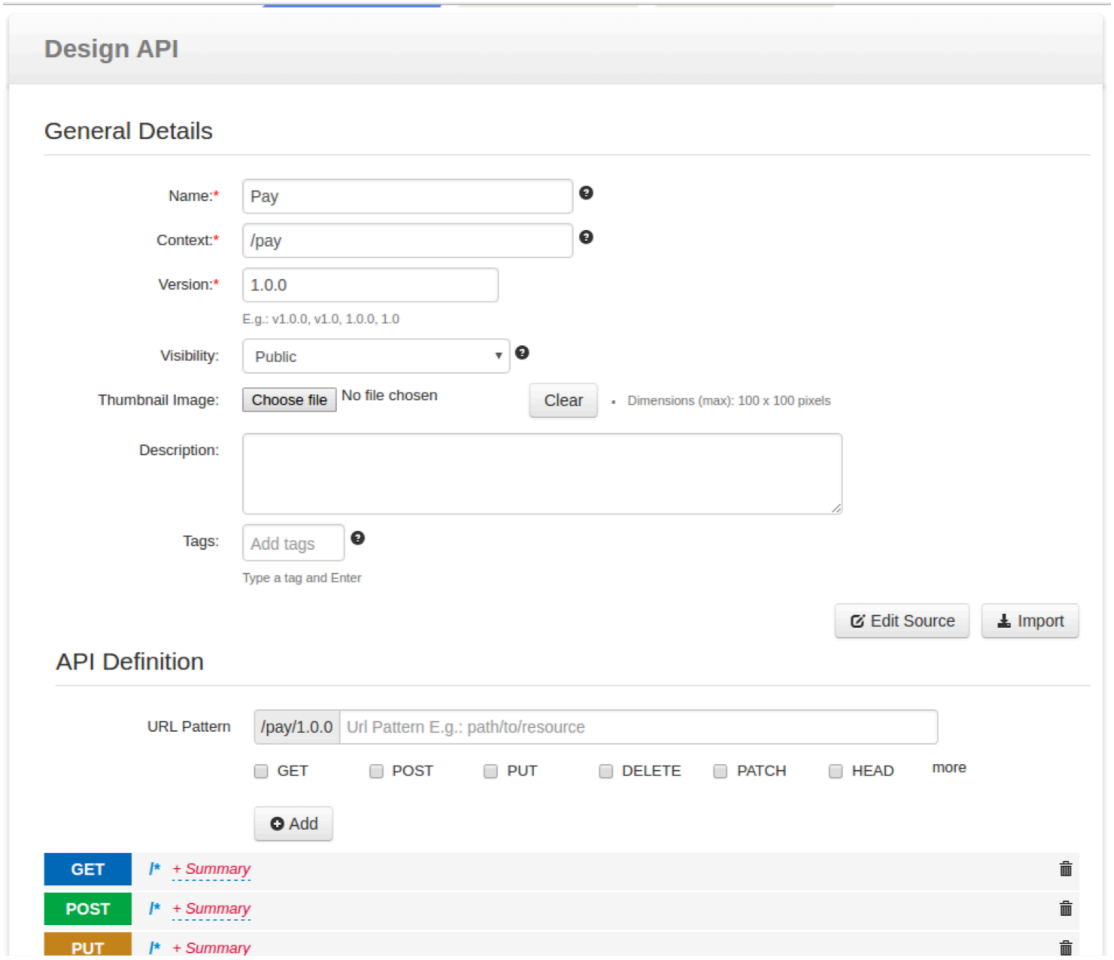
9) Click I have an Existing API

10) Do **not choose** either of the Swagger options, but click **Start Creating**

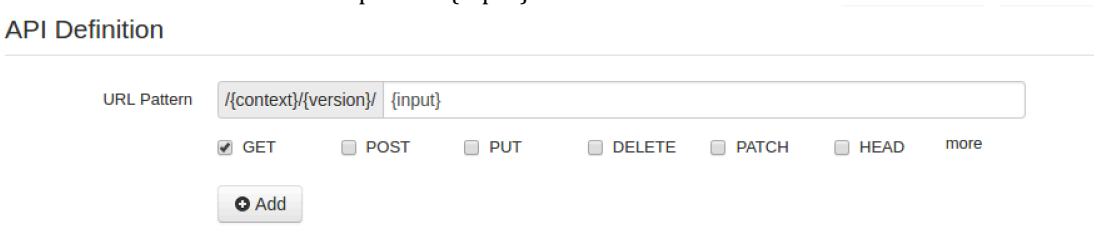


11) Use the following

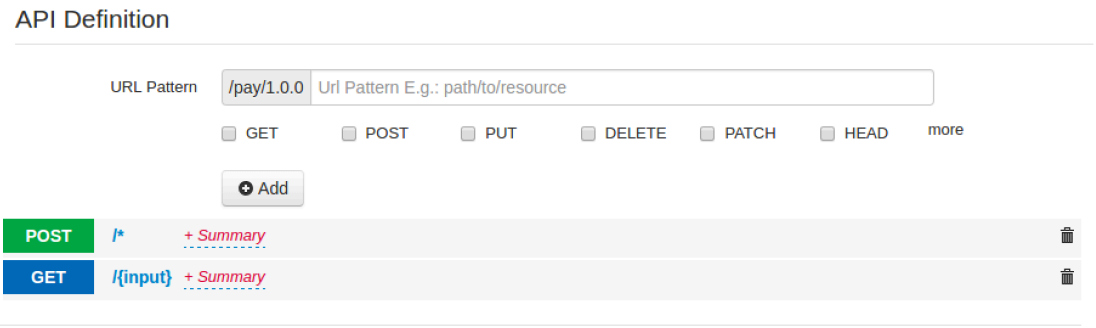
Field	Value	Description
Name	Pay	Name of API as you want it to appear in the API store
Context	/pay	URI context path that is used by to API consumers
Version	1.0.0	API version (in the form of version.major.minor)
Visibility	Public	You can require users to be authorized into a role or domain before they can see this API. We are making it fully visible: even to users who are not signed in.
Thumbnail Image	Your choice or leave blank	I like kittens :-)
Description	Up to you	
Tags	Again up to you	
Resources	Leave for the moment	This area allows you to define specific RESTful resources which can then have permissions applied, improved documentation, etc


Design API
General Details
Name:*
Pay
Context:*
/pay
Version:*
1.0.0
E.g.: v1.0.0, v1.0, 1.0.0, 1.0
Visibility:
Public
Thumbnail Image:
Choose file
No file chosen
Clear
Dimensions (max): 100 x 100 pixels
Description:
Tags:
Add tags
Type a tag and Enter
Edit Source
Import
API Definition
URL Pattern
/pay/1.0.0
Url Pattern E.g.: path/to/resource
GET
POST
PUT
DELETE
PATCH
HEAD
more
Add
GET
/*
+ Summary
POST
/*
+ Summary
PUT
/*
+ Summary

12) Now "shape" the API and define the verbs properly:
    a. Delete the GET, PUT and DELETE verbs
    b. Re-add the GET verb with a URL pattern {input}:

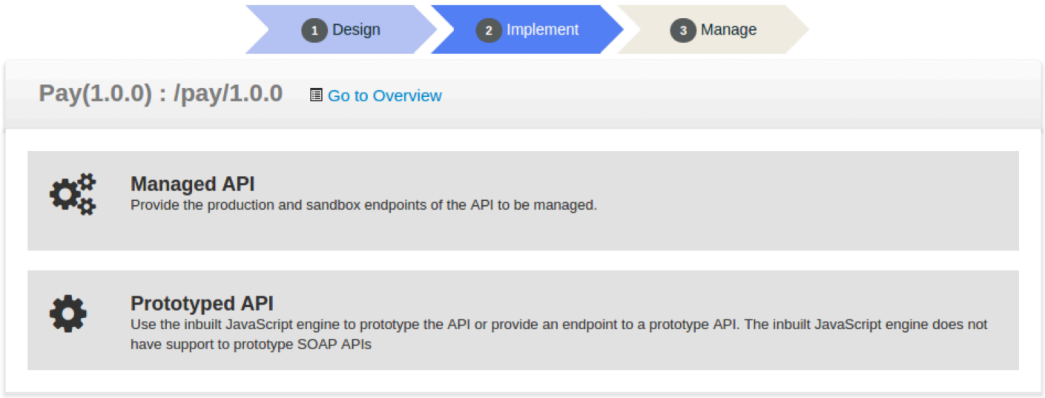


    c. It should now look like this:



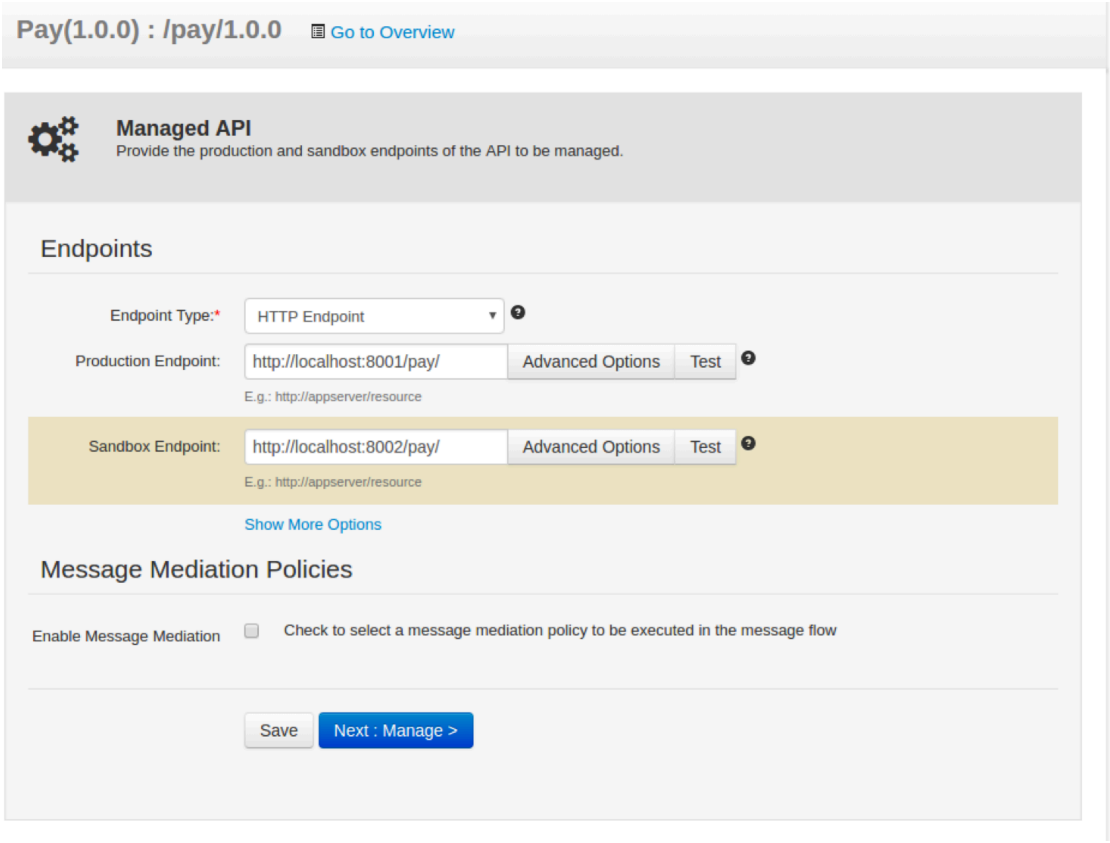
13) Now click **Implement**

14) We now want to choose Managed API:



15) Use the following information

Field	Value	Description
Endpoint type	HTTP endpoint	
Production endpoint	http://localhost:8001/pay/	This is the URL of the ESB plus SOAP service
Sandbox endpoint	http://localhost:8002/pay/	This is the URL of the Node.js sandbox service



11) Now Click **Manage** to go to the Manage tab and provide the following information:

Field	Value	Description
Default Version	Ticked	There can be a default version of every API, which can be routed to whichever version the administrator or API owner chooses.
Tier Availability	Select all of Bronze/Gold/Silver/Unlimited	The API can be available at different level of service; you can select multiple entries from the list. At subscription time, the consumer chooses which tier they are interested in.
Transports	HTTP/HTTPS	This allows users to use HTTP to access this URL. In practice this is a bad idea for a production system because the API Key (Bearer Token - which we'll see later) needs to be protected.
Enable Hard Throttling Limits	Tick then: Production Limit: 20 Sandbox Limit: 5	These enforce hard TPS limits against the service to protect the backend for DDoS attacks or simply if it isn't very scalable!
Response Caching	Disabled	This allows the API Manager to cache responses from the backend which can improve performance if there is cacheable content
Other fields	Take a look but leave the same	

16) Now click **Save.**

17) On the left hand menu click APIs->Browse.

18) Have a look at the API

19) **Adding Documentation:** After creating the API, click on its icon to open its details. Select the Docs tab.

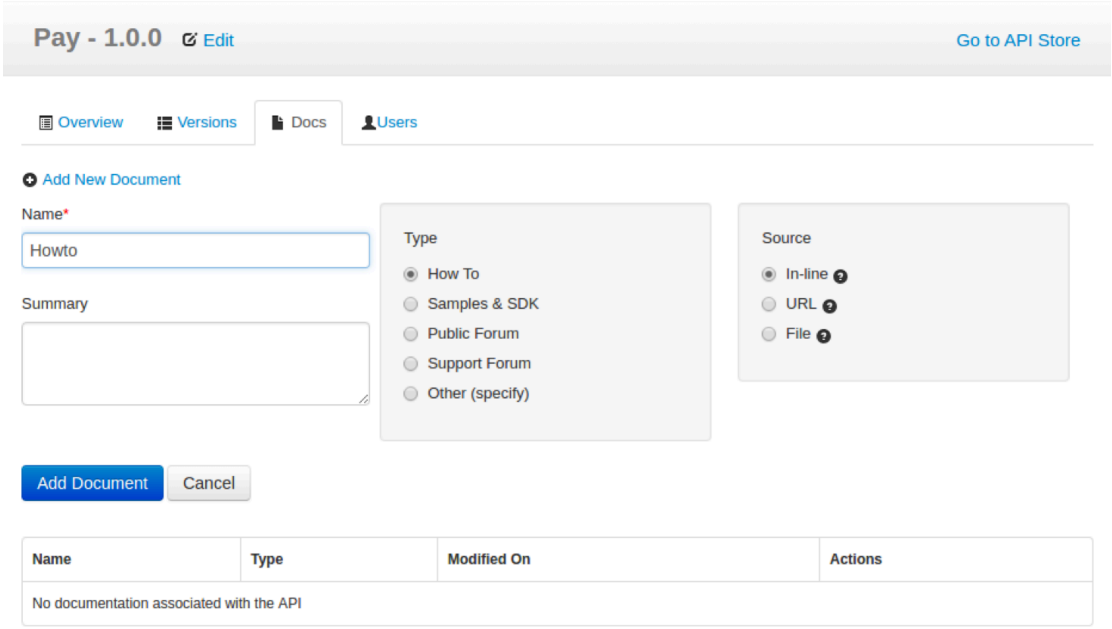
20) Click **Add New Document** link.

Documentation can be provided 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:

1. Swagger documents
2. How To
3. Samples and SDK
4. Public forum / Support forum (external link only)
5. API message formats
6. Other

21) Select the **How To** type, a name for the document and a short description, which will appear in the API Store. Select inline or provide a URL.

22) Click **Add Document**.



23) Once the document is added, click **Edit Content** link, which opens an embedded editor to edit the document contents.

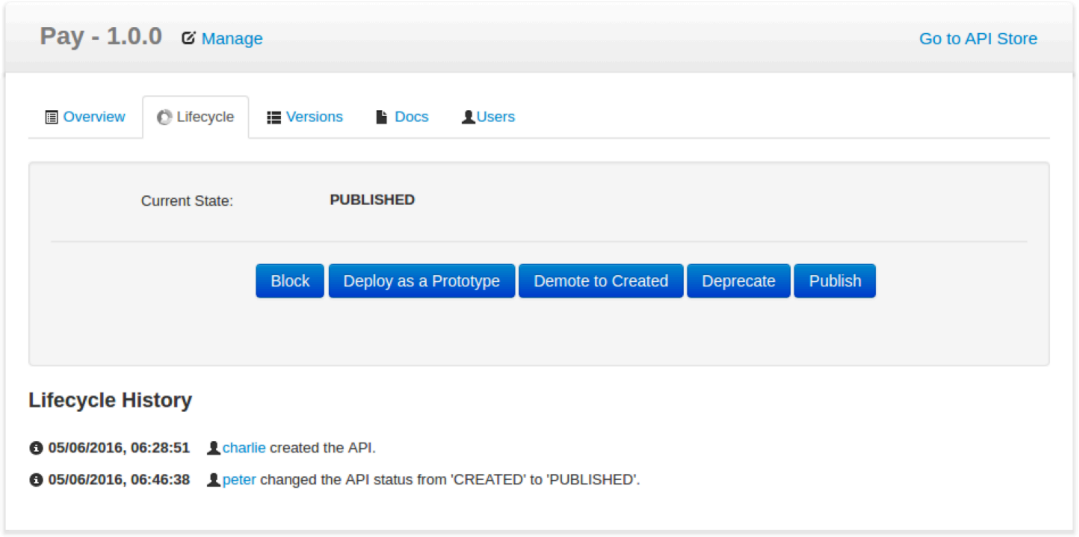
Publishing!

24) Log out as **Charlie**

25) Log in as **peter/password**

26) Now Click on the Pay Icon:

27) Choose the **Lifecycle** tab. Select **Publish**.



This is a useful page. You can see the audit log, the potential lifecycle/governance states of the service and more. We'll come back to this page in a bit and it will be even better!

Subscription

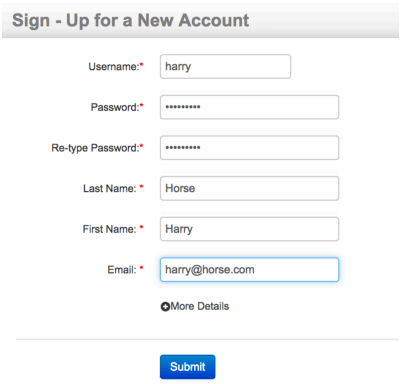
28) You subscribe to APIs using the API Store Web application

    Open the API Store (https://localhost:9447/store) using your browser. Using the API Store, you can,

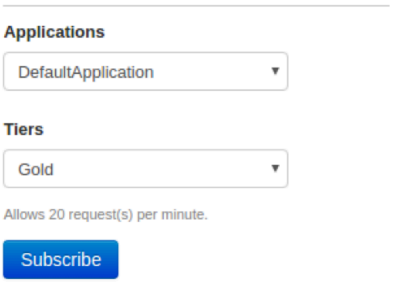
    - Search and browse APIs
    - Read documentation
    - Subscribe to APIs
    - Comment on, rate and share/advertize APIs
    - Take part in forums and request features etc.

The API you published earlier is available in the API Store.

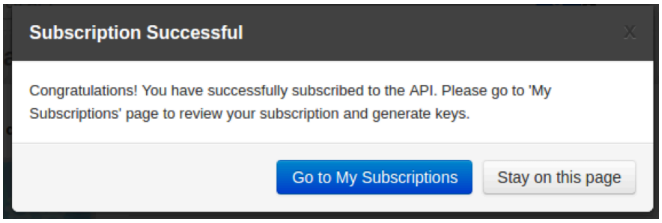
29) Self sign up to the API Store using the **Sign-up** link. You can use any appropriate (or even inappropriate) details.



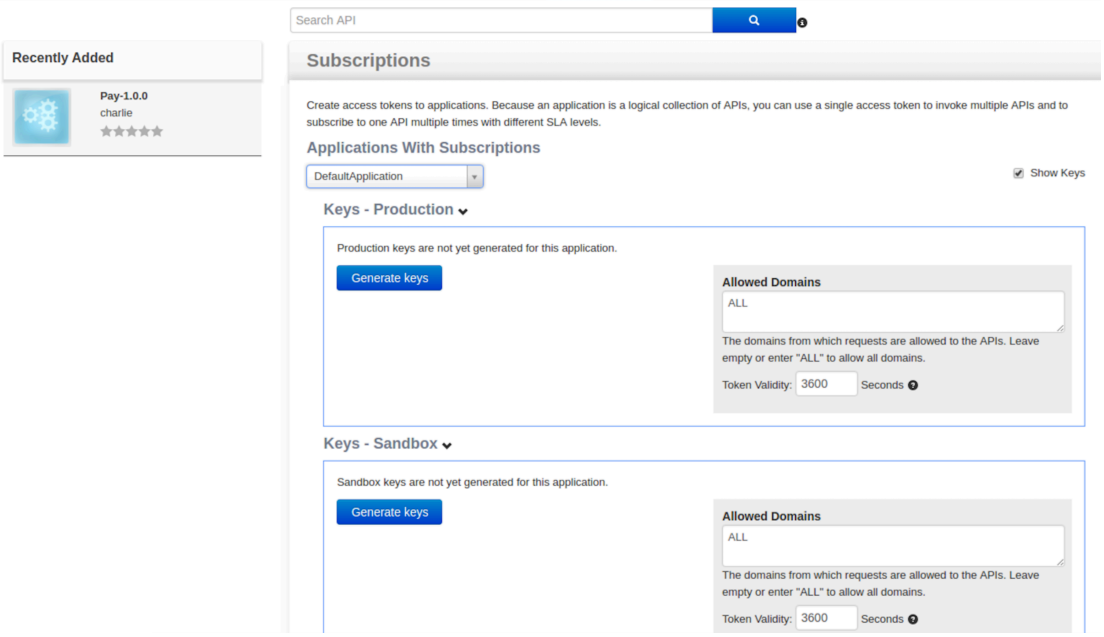
30) After signup, log in to the API Store and click the API that "peter" published earlier (Pay 1.0.0).
31) Note that you can see the subscription option in the right hand side of the UI after logging in. Select the default application, the Gold tier and click **Subscribe**.



You should see:

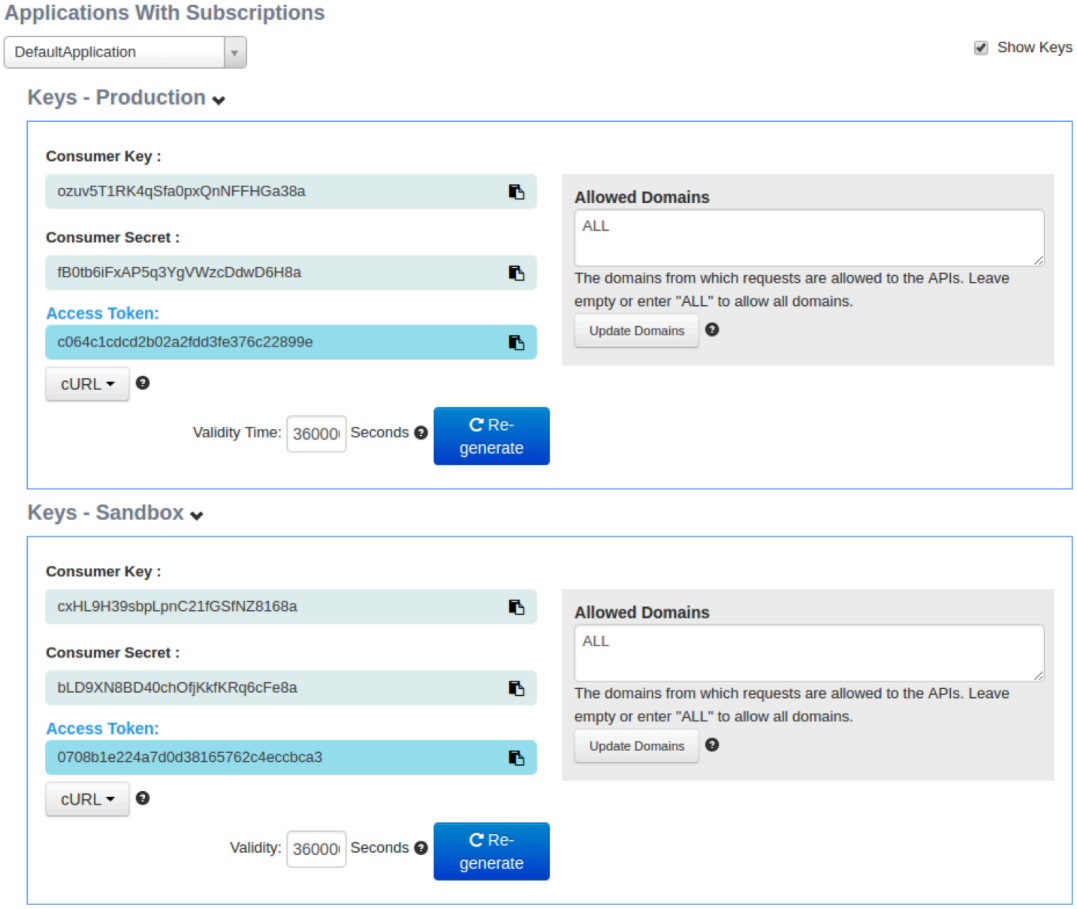


Click **Go to My Subscriptions**
It should look like:



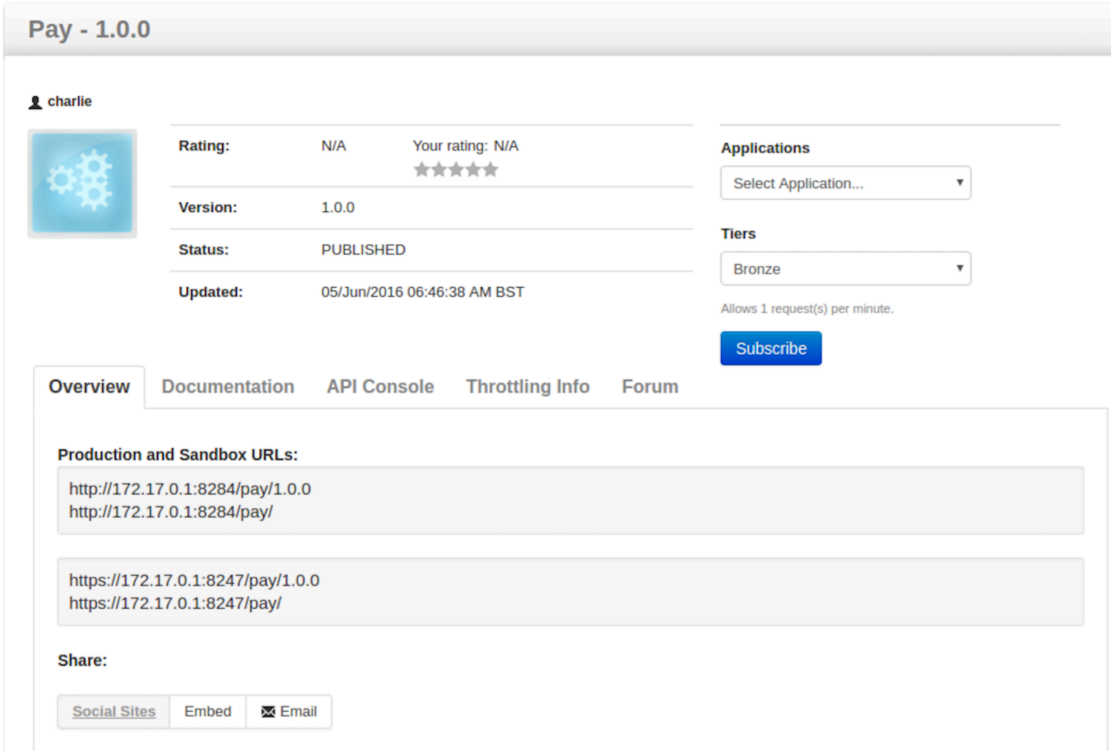
You can see that you are subscribed to the API.

32) Click **Generate** to create both a production and sandbox key. I recommend extending the validity period (add a few 00s to the end).



You can now test your API.

33) The easiest is the API Console. Click **APIs.** Click **Pay**.



Click **API Console**

34) Click on **GET**

35) Add a simple string as the required input.

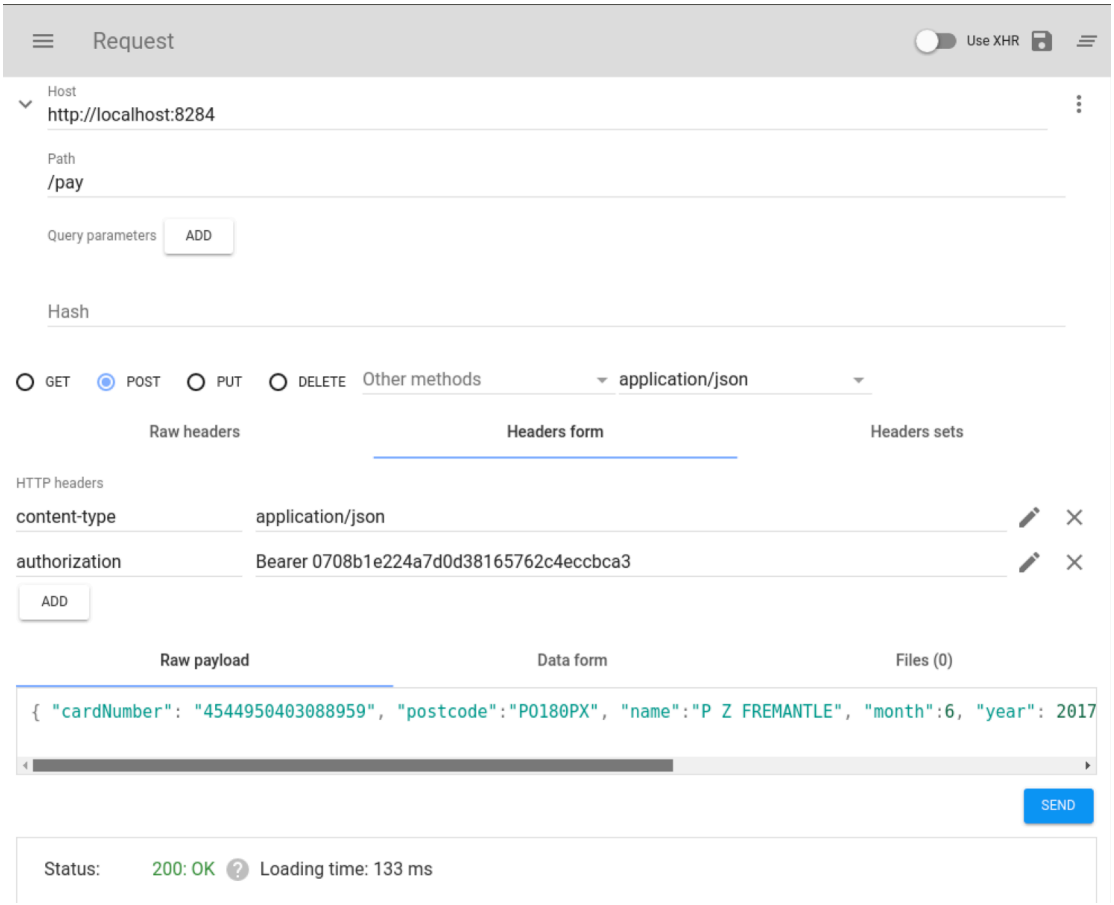
36) Click Try It Out.

37) We don't have a good enough Swagger definition to use the API Console to test the POST method. We would need to have created one and then imported into the API Manager as part of the creation stage. However, we can go back to the Advanced REST Client to test the POST method.

38) In the Advanced REST Client, use the following:

Host: http://localhost:8284
Path /pay/
POST
Headers: Content-Type: application/json
Authorization: Bearer <your production bearer token here>
Payload:

```
{
  "cardNumber": "4544950403888999",
  "postcode": "PO107XA",
  "name": "P Z FREMANTLE",
  "month": 6,
  "year": 2017,
  "cvc": "999",
  "merchant": "A0001",
  "reference": "test",
  "amount": 11.11
}
```



Also try out hitting multiple times (to see throttling), and also try the sandbox token instead.
How can you tell that the Sandbox token has worked?

39) **Analytics**

Analytics are not yet enabled on the server.

In a fresh Terminal window, start up the WSO2 Data Analytics Server 3.0.1
cd ~/servers/wso2das-3.0.1
bin/wso2server.sh

You can take a look at the DAS Admin Console at:
https://localhost:9448

Now go to the API Admin Dashboard:
https://localhost:9447/admin-dashboard

Login as **admin/admin**

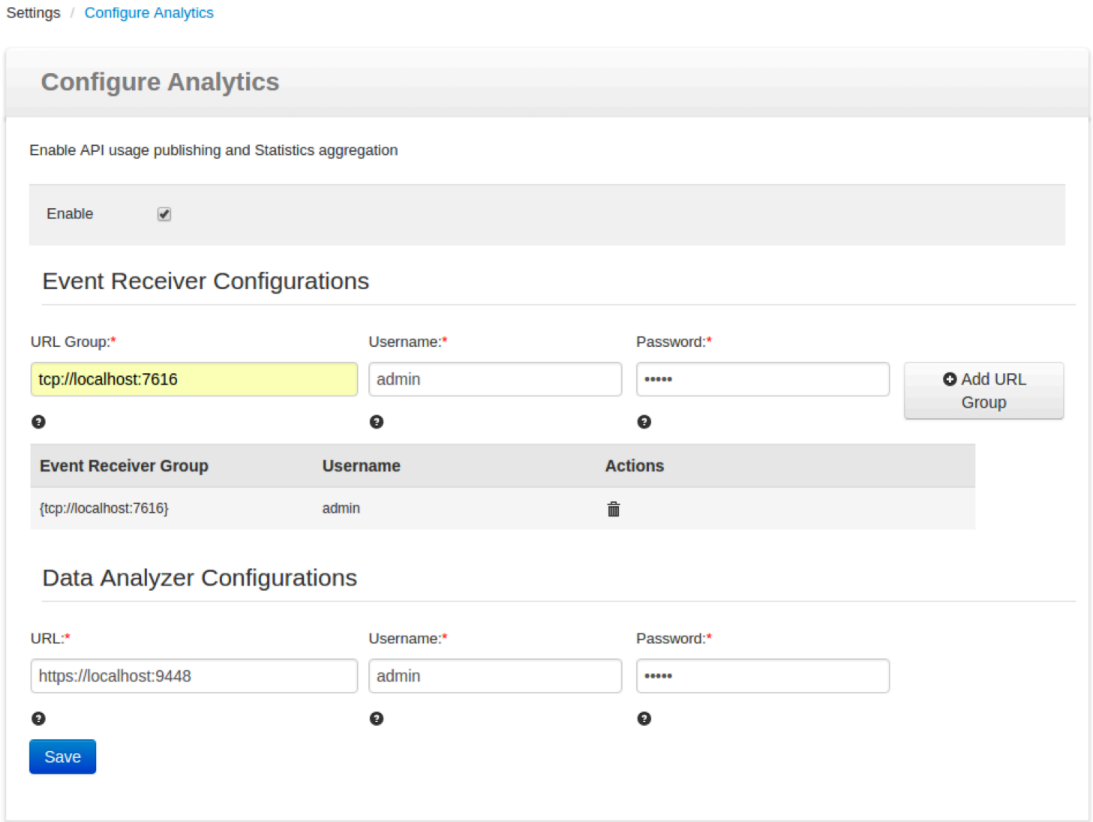
Click on **Settings/Configure Analytics**

Change the URL Group URI to be:
`tcp://localhost:7616 admin admin`

Click Add URL Group.

Change the Data Analyzer Configurations to be
`https://localhost:9448 admin admin`

Click **Save**

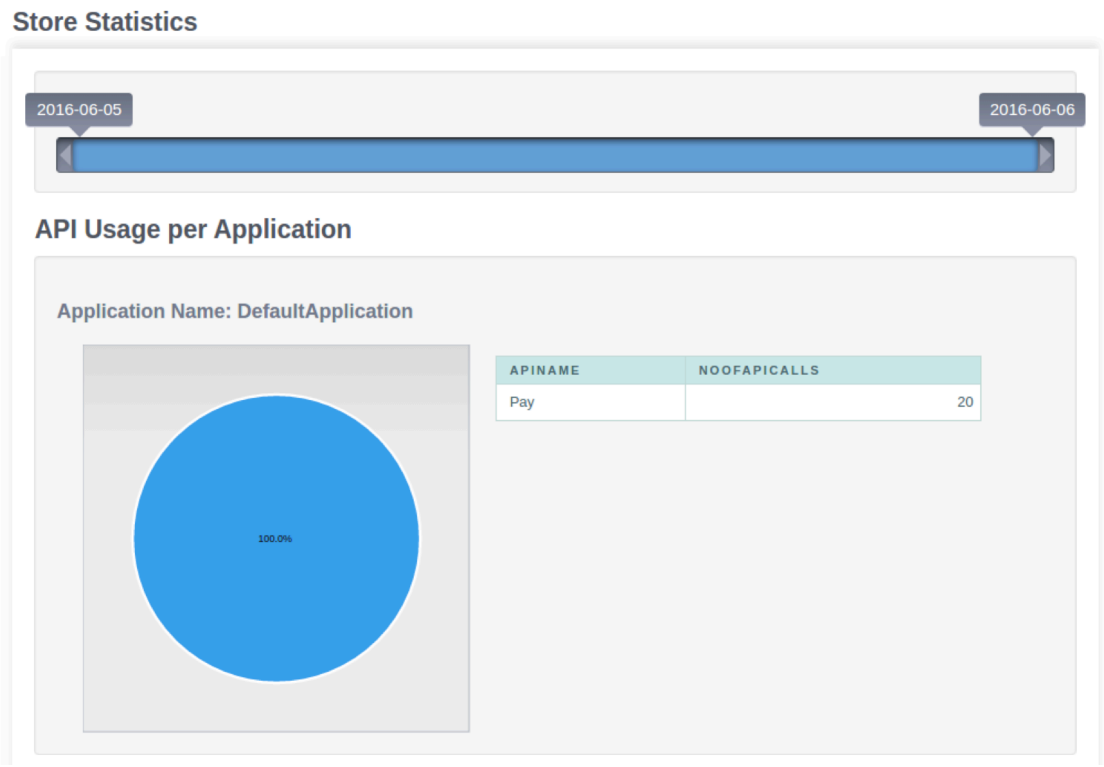


Now go back to ARC and generate some traffic. You also need to wait 5 minutes for the analytics batch to run. Maybe it's a good time for a tea or coffee!

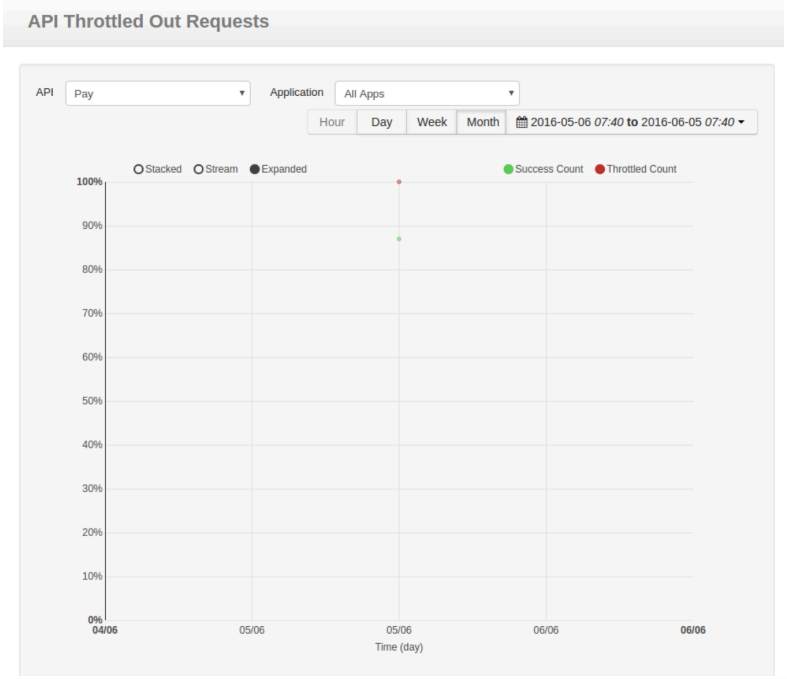
You can see statistics both as a user and as a publisher.

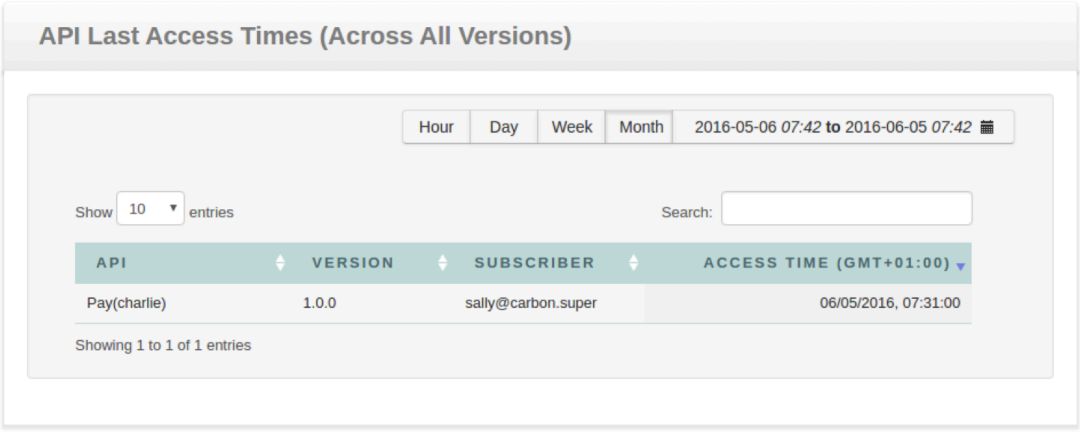
First go to the store (https://localhost:9447/store)

While you are still in the store, look at the Statistics.



40) Log back into the publisher and take a look at the stats there as well.





41) **Extras**

If you are finished early, there are lots more things to try. For example, see if you can Copy your existing API into a new version and publish that.

Send some requests into that and now check out the statistics.

That's all folks!

Appendix A

Setting up 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 API Gateway) 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 (i.e., 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 lifecycle and monetization aspects. The publisher is also interested in usage patterns for APIs and as such 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. S/he subscribes to APIs to obtain API keys.

The admin user can play the creator, publisher and subscriber roles described earlier. In this section, we explain how to set up these users or custom users and roles.

1. Log in to the management console user interface (https://apimgr:9447/carbon) of the API Manager using admin/admin credentials.
2. Select the **Users and Roles** menu under the **Configure** menu.
3. Click **Add New Role** and provide creator as the role name.
4. Click **Next**.
5. Select the following permissions from the list that opens and then click **Finish**.
    - 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. You can now create users for each of those roles. To do so, click the **Users and Roles** menu under the **Configure** menu.
8. Click **Users**.
9. Click **Add New User**, provide the username/password and click **Next**.
10. Select the role you want to assign to the user (e.g., creator, publisher or subscriber) and click **Finish**. Given below is a list of usernames and the roles we assign to them in this guide.

11. Repeat the steps to create at least one user for all roles.