

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.

Let's get started!

- I have an Existing API**
Use an existing API's endpoint or the API definition to create an API.
- I have a SOAP Endpoint**
Use an existing SOAP endpoint to create a managed API. Import WSDL of the SOAP service.
- Design new API**
Design and prototype a new API.

- 9) Click I have an Existing API

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

I have an Existing API
Use an existing API's endpoint or the API definition to create an API.

Import the API definition as **Swagger file** **Swagger URL**

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 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

API Definition

URL Pattern /pay/1.0.0 Url Pattern E.g.: path/to/resource

GET POST PUT DELETE PATCH HEAD more

GET	/* + Summary	<input type="button" value="Delete"/>
POST	/* + Summary	<input type="button" value="Delete"/>
PUT	/* + Summary	<input type="button" value="Delete"/>

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}:

API Definition

URL Pattern

GET POST PUT DELETE PATCH HEAD more

[+ Add](#)

- c. It should now look like this:

API Definition

URL Pattern Url Pattern E.g.: path/to/resource

GET POST PUT DELETE PATCH HEAD more

[+ Add](#)

POST	/*	+ Summary	
GET	/{{input}}	+ Summary	

13) Now click **Implement**

14) We now want to choose Managed API:

1 Design 2 Implement 3 Manage

Pay(1.0.0) : /pay/1.0.0 [Go to Overview](#)

Managed API
Provide the production and sandbox endpoints of the API to be managed.

Prototyped API
Use the inbuilt JavaScript engine to prototype the API or provide an endpoint to a prototype API. The inbuilt JavaScript engine does not have support to prototype SOAP APIs

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

The screenshot shows the 'Endpoints' configuration for the 'Pay' API. It includes fields for Production Endpoint (http://localhost:8001/pay/) and Sandbox Endpoint (http://localhost:8002/pay/), both set to HTTP Endpoint type. There are also 'Advanced Options' and 'Test' buttons. A 'Message Mediation Policies' section with an enable checkbox is present, along with 'Save' and 'Next : Manage >' buttons.

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**.

The screenshot shows the 'Add Document' form for a version labeled 'Pay - 1.0.0'. At the top, there are tabs for 'Overview', 'Versions', 'Docs' (which is selected), and 'Users'. A 'Go to API Store' button is also present. The main form fields include:

- Name***: A text input field containing 'Howto'.
- Summary**: An empty text area.
- Type**: A group of radio buttons:
 - How To
 - Samples & SDK
 - Public Forum
 - Support Forum
 - Other (specify)
- Source**: A group of radio buttons:
 - In-line
 - URL
 - File

At the bottom of the form are 'Add Document' and 'Cancel' buttons. Below the form, a table displays documentation details:

Name	Type	Modified On	Actions
No documentation associated with the API			

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**.

The screenshot shows a web interface for managing an API named "Pay - 1.0.0". At the top, there are tabs for Overview, Lifecycle (which is selected), Versions, Docs, and Users. Below the tabs, it says "Current State: PUBLISHED". There are several buttons at the bottom: Block, Deploy as a Prototype, Demote to Created, Deprecate, and Publish. Under the Lifecycle tab, there is a section titled "Lifecycle History" with two entries:

- 05/06/2016, 06:28:51 charlie created the API.
- 05/06/2016, 06:46:38 peter changed the API status from 'CREATED' to 'PUBLISHED'.

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/advertise 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.

The screenshot shows a "Sign - Up for a New Account" form. It has fields for Username (harry), Password, Re-type Password, Last Name (Horse), First Name (Harry), and Email (harry@horse.com). There is also a "More Details" link and a "Submit" button.

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**.

Applications

DefaultApplication ▾

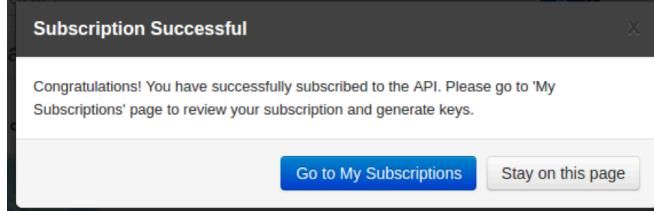
Tiers

Gold ▾

Allows 20 request(s) per minute.

Subscribe

You should see:



Click **Go to My Subscriptions**

It should look like:

Search API 🔍 ⚙️

Recently Added

Pay-1.0.0
charlie
★★★★★

Subscriptions

Create access tokens to applications. Because an application is a logical collection of APIs, you can use a single access token to invoke multiple APIs and to subscribe to one API multiple times with different SLA levels.

Applications With Subscriptions

DefaultApplication ▾
 Show Keys

Keys - Production ▾

Production keys are not yet generated for this application.

Generate keys

Allowed Domains
ALL

The domains from which requests are allowed to the APIs. Leave empty or enter "ALL" to allow all domains.

Token Validity: 3600 Seconds ⓘ

Keys - Sandbox ▾

Sandbox keys are not yet generated for this application.

Generate keys

Allowed Domains
ALL

The domains from which requests are allowed to the APIs. Leave empty or enter "ALL" to allow all domains.

Token Validity: 3600 Seconds ⓘ

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).

Applications With Subscriptions

DefaultApplication Show Keys

Keys - Production ▾

Consumer Key : `ozuv5T1RK4qSfa0pxQnNFFHGa38a`

Consumer Secret : `fB0tb6IFxAP5q3YgVWzcDdwD6H8a`

Access Token: `c064c1cdcd2b02a2fdd3fe376c22899e`

cURL

Validity Time: Seconds **C Re-generate**

Allowed Domains
ALL

The domains from which requests are allowed to the APIs. Leave empty or enter "ALL" to allow all domains.

Update Domains

Keys - Sandbox ▾

Consumer Key : `cxHL9H39sbpLpnC21fGSfNZ8168a`

Consumer Secret : `bLD9XN8BD40chOfjKkfKRq6cFe8a`

Access Token: `0708b1e224a7d0d38165762c4eccbca3`

cURL

Validity: Seconds **C Re-generate**

Allowed Domains
ALL

The domains from which requests are allowed to the APIs. Leave empty or enter "ALL" to allow all domains.

Update Domains

You can now test your API.

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

The screenshot shows the API Manager interface for the 'Pay' API version 1.0.0. At the top, there's a header with the API name and version. Below it, a user profile for 'charlie' is shown. To the right, there are sections for 'Applications' (with a dropdown menu), 'Tiers' (set to 'Bronze'), and a note about request limits. A 'Subscribe' button is also present. Below these, tabs for 'Overview', 'Documentation', 'API Console', 'Throttling Info', and 'Forum' are visible, with 'Overview' being the active tab. Under the 'Overview' tab, there's a section for 'Production and Sandbox URLs' containing two links: <http://172.17.0.1:8284/pay/1.0.0> and <http://172.17.0.1:8284/pay/>. Further down, there's a 'Share:' section with options for 'Social Sites', 'Embed', and 'Email'.

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  
}
```

The screenshot shows the 'Request' tab of the Advanced REST Client. The 'Host' field contains 'http://localhost:8284'. The 'Path' field contains '/pay'. The 'Method' dropdown is set to 'POST'. The 'Content-Type' dropdown is set to 'application/json'. The 'Headers form' section shows 'content-type: application/json' and 'authorization: Bearer 0708b1e224a7d0d38165762c4eccbca3'. The 'Raw payload' section contains the JSON payload from the previous code block. The status bar at the bottom shows 'Status: 200: OK' and 'Loading time: 133 ms'.

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**

Settings / [Configure Analytics](#)

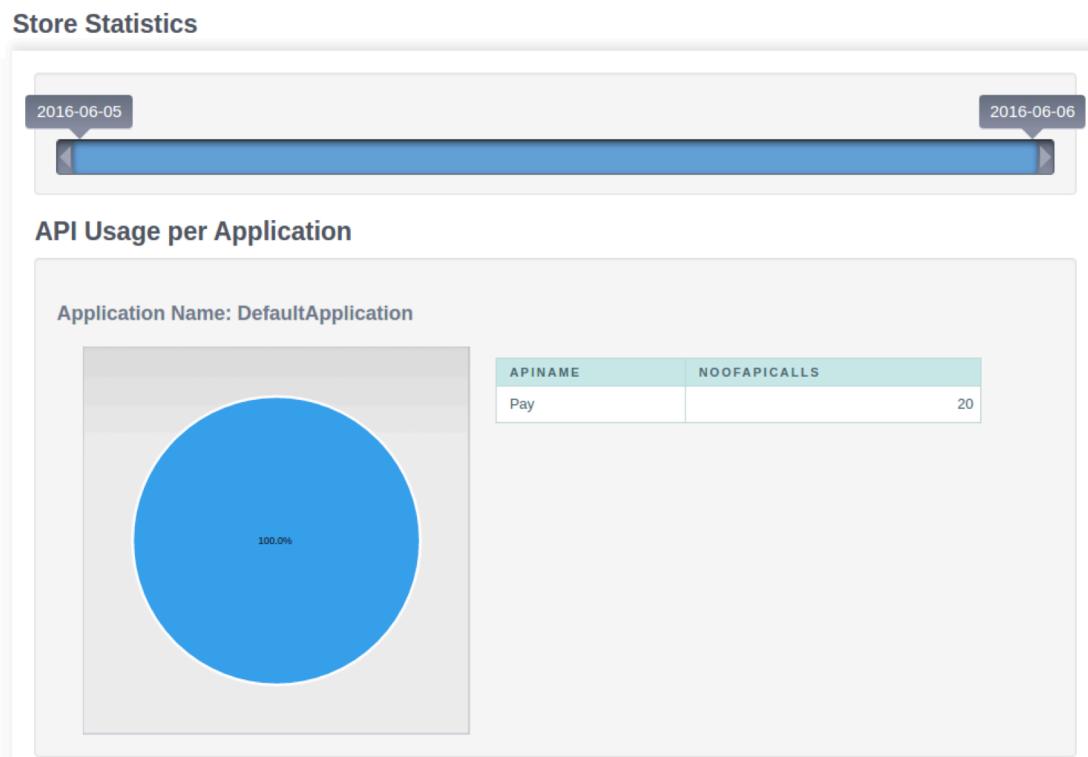
The screenshot shows the 'Configure Analytics' page. At the top, there's a note: 'Enable API usage publishing and Statistics aggregation'. Below it, under 'Event Receiver Configurations', there's a table with one row. The first column is 'URL Group:' with value 'tcp://localhost:7616' highlighted in yellow. The second column is 'Username:' with value 'admin'. The third column is 'Password:' with value '*****'. To the right of the table is a button labeled '+ Add URL Group'. Below this section is another table for 'Event Receiver Group' with one row: '{tcp://localhost:7616}' in the 'Event Receiver Group' column, 'admin' in the 'Username' column, and a trash icon in the 'Actions' column. Under 'Data Analyzer Configurations', there's a similar table with one row: 'URL:' with value 'https://localhost:9448', 'Username:' with value 'admin', and 'Password:' with value '*****'. A blue 'Save' button is located at the bottom left of this section.

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!

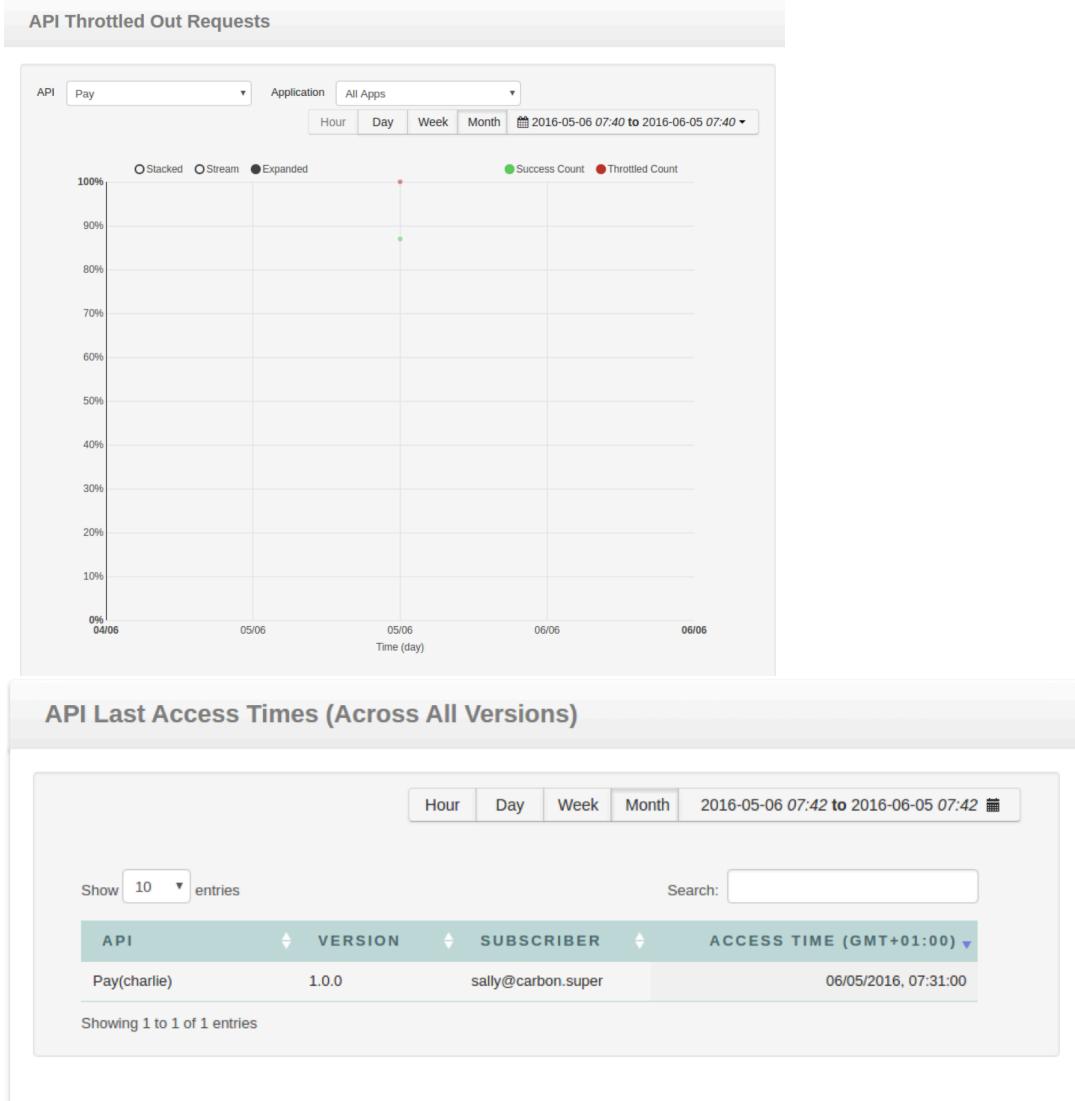
You can see statistics both as a user and as a publisher.

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.