Exercise 14

Messaging and MQTT, Wire Tap pattern, MongoDB

Pre-reqs
ESB configuration from Exercise 10

Objectives
In this lab we are going to explore some alternatives to what we've covered so far.

We are going to use a message pub/sub protocol (MQTT) and store messages into a NoSQL database MongoDB. The plan is to use the Wire Tap pattern to grab the messages as they flow through the ESB and then pass them over to a pub/sub broker. Once they are there we can easily subscribe to them and store them in the Database.

Install MQTT sender support into the WSO2 ESB

1. Before we can implement the wire tap, we need to enable the ability to send MQTT messages in the WSO2 ESB. Let's add the required libraries.

   Download the following two files (in each case, follow the link and then click **View Raw** to start the download).
   http://freo.me/mqtt-jar
   http://freo.me/mqtt-axis2

   Now (each cp is all one line!):
   ```
   cp ~/Downloads/mqtt-client-0.4.0.jar ~/servers/wso2esb-4.9.0/repository/components/lib

   cp ~/Downloads/ axis2-transport-mqtt-1.0.0.jar ~/servers/wso2esb-4.9.0/repository/components/lib
   ```

2. Edit axis2.xml:
   ```
   atom ~/servers/wso2esb-4.9.0/repository/conf/axis2/axis2.xml
   ```

   On line 453 remove the comment marks <!-- and --> so that the MQTT transport sender is enabled.

3. Restart the ESB (or start it) the same as in Exercise 10.
   cd ~/servers/wso2esb-4.9.0/
   bin/wso2server.sh

4. **Install Mosquitto**

   ```
   sudo apt-get install -y mosquitto mosquitto-clients
   ```

5. This will automatically start mosquitto but I'd rather see the verbose debug, so:

   ```
   sudo service mosquitto stop
   mosquitto -v
   ```

6. I have updated the PayAPI from Exercise 10 to add the wiretap code.

7. You can deploy this by downloading the .car file from:
   http://freo.me/pay-wiretap

   Click View Raw to download it.

8. You can deploy this API CAR file via the console, or you can simply copy it into the right place:
   `cp ~/Downloads/WireTapProject_1.0.0.car ~/server/wso2esb-4.9.0/repository/deployment/server/carbonapps`

9. You can look at the changes from the previous API in the console.
   The main change is the addition of this mediation XML *before* we convert the payload body to XML.

```
<clone continueParent="true">
  <target>
        <sequence>
          <call>
                <endpoint>
                  <address
                           uri=

"mqtt:/local?mqtt.server.host.name=localhost&amp;mqtt.server.port=1883&amp;mqtt.client.id=esb.test.sender&amp;mqtt.topic.name=/pay&amp;mqtt.subscription.qos=0&amp;mqtt.blocking.sender=true" />

                </endpoint>

          </call>
        </sequence>
  </target>
</clone>
```

10. Make sure the backend is running.
    If not then you can do:
    `sudo docker run -d -p 8080:8080 pizak/pay`

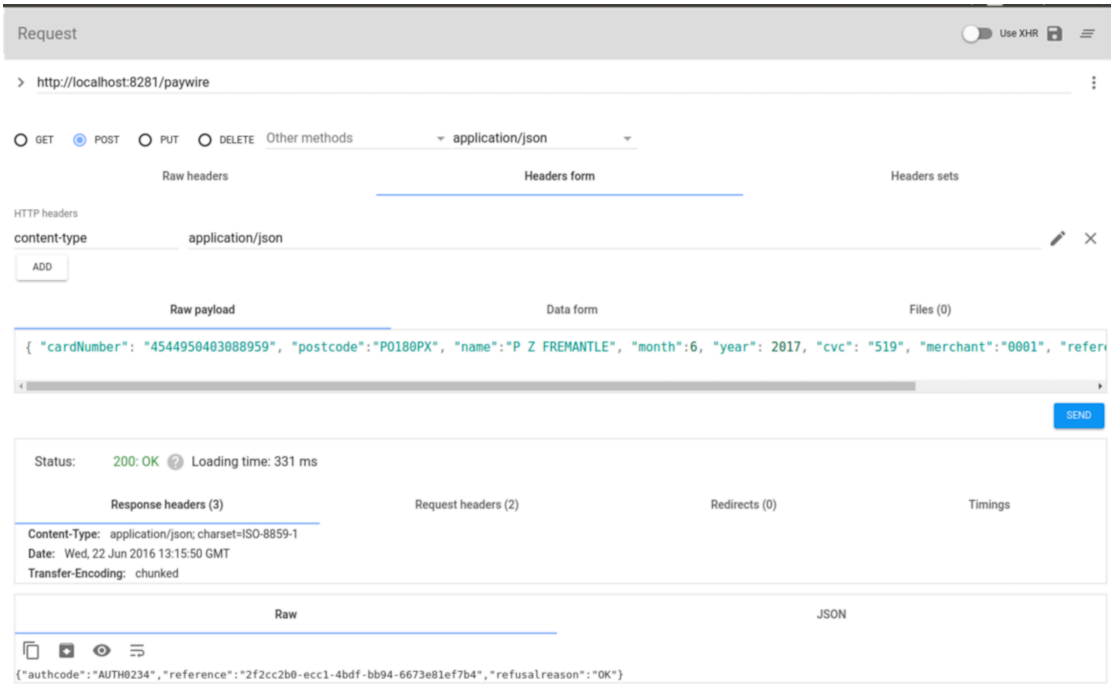
11. Start a subscriber to MQTT in a new terminal window so you can see any messages to the MQTT server:

    ```
    mosquitto_sub -v -t  "/#"
    ```

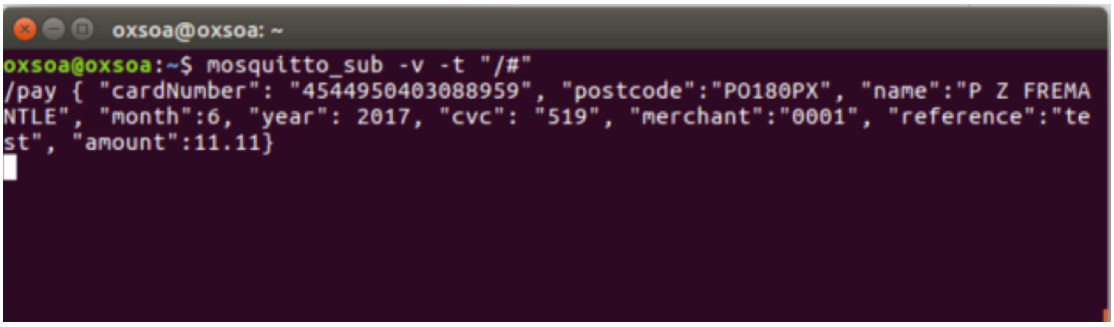
    If you look at the mosquitto server window you should see this happen.

This subscribes to all topics and uses a verbose output.

12. Now try out the API using the Advanced REST client
    Everything is the same as in Exercise 10, except that the URI is
    http://localhost:8281/paywire



13. You should see a copy of the JSON now appear in your subscriber window:



14. Part B - storing the messages from the wiretap into MongoDB

15. We are going to write a very simple Node.js app to grab the messages from the MQTT broker and save them into MongoDB.

16. Firstly we need some dependencies installed:
```
sudo apt-get install -y mongodb
```

17. Next we need some libraries for Node.js
```
npm install mqtt mongodb uuid
```

18. Here is a really simple node app which will listen on MQTT /pay and save those messages to Mongo:

```
var mqttClient    = require('mqtt').connect('mqtt://localhost');
var uuid = require('uuid');

mqttClient.on('connect', function () {
  mqttClient.subscribe('/pay', {qos:0});
});

var mongoClient =
require('mongodb').MongoClient.connect("mongodb://localhost/test",
function(err, db) {
      if (err==null) {
            mqttClient.on('message', function (topic, message) {
      json = JSON.parse(message);
      json.uuid = uuid.v4();
                  insertJSON(db, 'pay', json, function(result) {
console.log(result);});
            });
      }
});
```

http://freo.me/ex14-mqtt

19. Create a file mqtt-mongo.js and copy this code into it.
20. Run this:
```
node mqtt-mongo.js
```

21. Now push some messages through the ESB and see them appear in Mongo.

22. You can look at the mongo database by doing: