Step by step information on creating an API pointing to a JAX-RS service and invoking it through API Console

Version: 2.0

Date: 13-August-2015

Revision History

Version	Date	Desc	Author
1.0	12-August-2015	Initial document explaining steps on how to invoke a JAX-RS service	Evanthika Amarasiri
2.0	13-October-2015	Added clear instructions on deploying the JAX-RS service in API Manager itself and invoking it	Evanthika Amarasiri

Table of Contents

1. **Deploying the JAX-RS service in API Manager**

   There are two ways you can do this. One is by dropping the .war file to the relevant deployment folder and the other is through the API Manager Management Console.

   Dropping the .war file manually

   - Drop the jaxrs_basic.war file to wso2am-1.9.1/repository/deployment/server/webapps.
   - Once successfully deployed, it would print the below message

   ```
   [2015-10-13 13:39:53,788]  INFO - TomcatGenericWebappsDeployer Deployed webapp: StandardEngine[Catalina].StandardHost[localhost].StandardContext[/jaxrs_basic].File[/WSO2/TestCases/Virtusa-Tool/RC4/wso2am-1.9.1/repository/deployment/server/webapps/jaxrs_basic.war]
   ```

   Upload through API Manager Management Console

   - Login to the Management Console
   - Navigate to **Home > Manage > Applications > Add > JAX-WS/JAX-RS**
   - Click on Browse and select the jaxrs-basic.war file
   - Once uploaded, view the webapp through **Home > Manage > Applications > List**
   - Click on **Find Services** link of the jaxrs-basic webapp from the list & you will be able to find the endpoint & ?wadl details.

   Endpoint address: http://localhost:9763/jaxrs_basic/services/customers
   WADL : http://localhost:9763/jaxrs_basic/services/customers?_wadl

2. Creating the API

Step 1

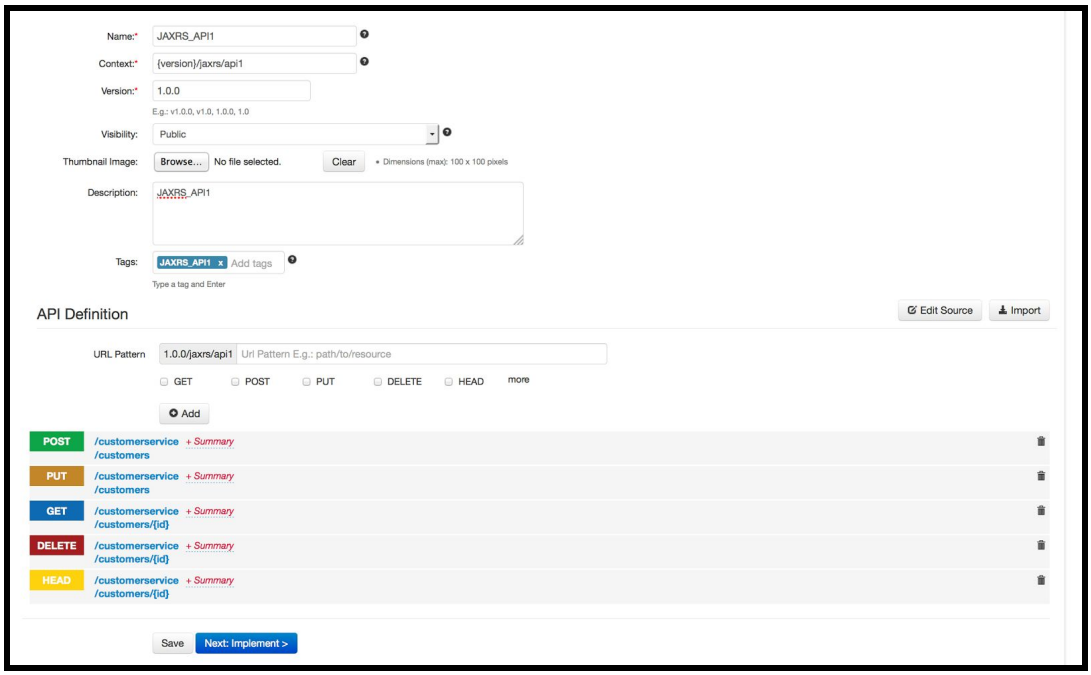
- Login to the API Publisher
- Click on **Add** and then select the radio button **Design new API**
- Click on the button **Start Creating** and you will be taken to the **Design API** page

Step 2

- Create an API with the following details

  Design section

  - **Name** : JAXRS_API1
  - **Context** : {version}/jaxrs/api1
  - **Version** : 1.0.0
  - **URL Pattern** :
    - Type in the pattern /customerservice/customers, tick the check boxes for **POST** & **PUT** & click on the button **Add**
    - Type in the pattern /customerservice/customers/{id}, tick the check boxes for **GET**, **DELETE** & **HEAD** & click on the button **Add**



Screenshot 1

Implement section

- **Endpoint Type** : HTTP Endpoint
- **Production Endpoint** :
  http://[hostname/ip]:[port]/jaxrs_basic/services/customers (Assuming that you have deployed it in API Manager, the URL would be something like http://localhost:9763/jaxrs_basic/services/customers)

Manage section

- **Tier Availability** : Unlimited
- Leave the rest of the values as they are for the rest of the fields and click 'Save & Publish'

3. Subscribe to an Application

Go to API Store, login and subscribe the API that you created to an application (Can be either an application which you created or the Default application)

4. Generate keys

- Go to My Subscriptions from the API Store
- Select the application which you subscribed your API to
- Click on **Generate Keys** button

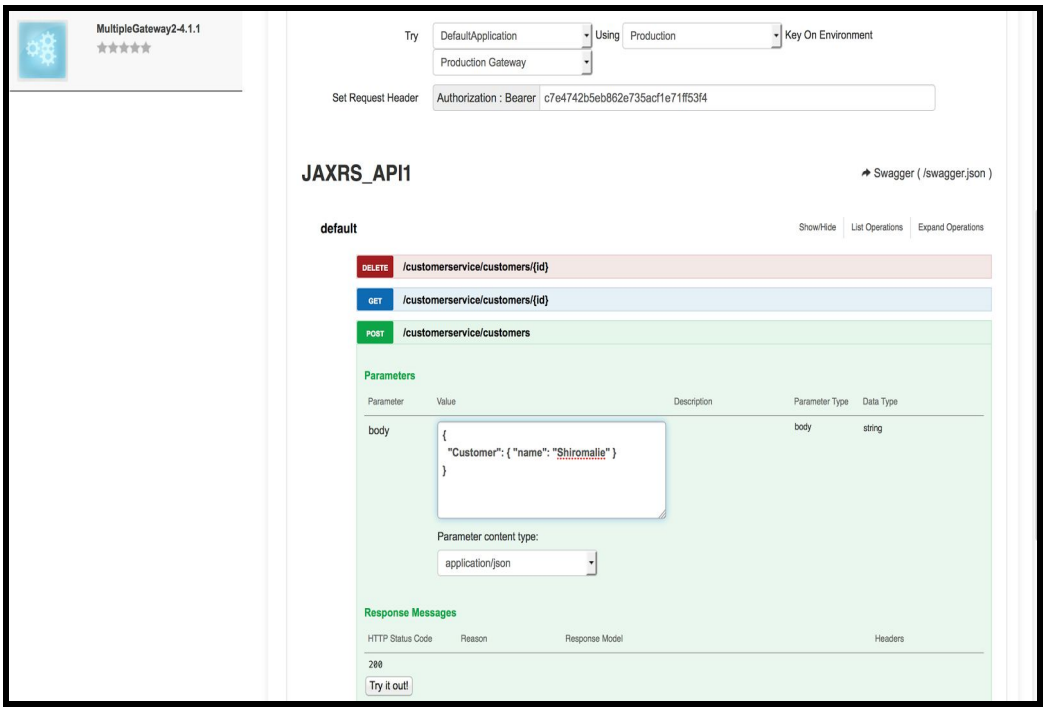
5. **Invoke** through API Console

You can invoke each resource method by inserting the required parameter value as shown below.

Step 1 **-** Invoke the POST resource

- For POST method you have to provide the post body

```
{
  "Customer": { "name": "Shiromalie" }
}
```



Screenshot 2

Response of POST

When you click on **Try It Now** button, you should receive a response as below.

Response Body

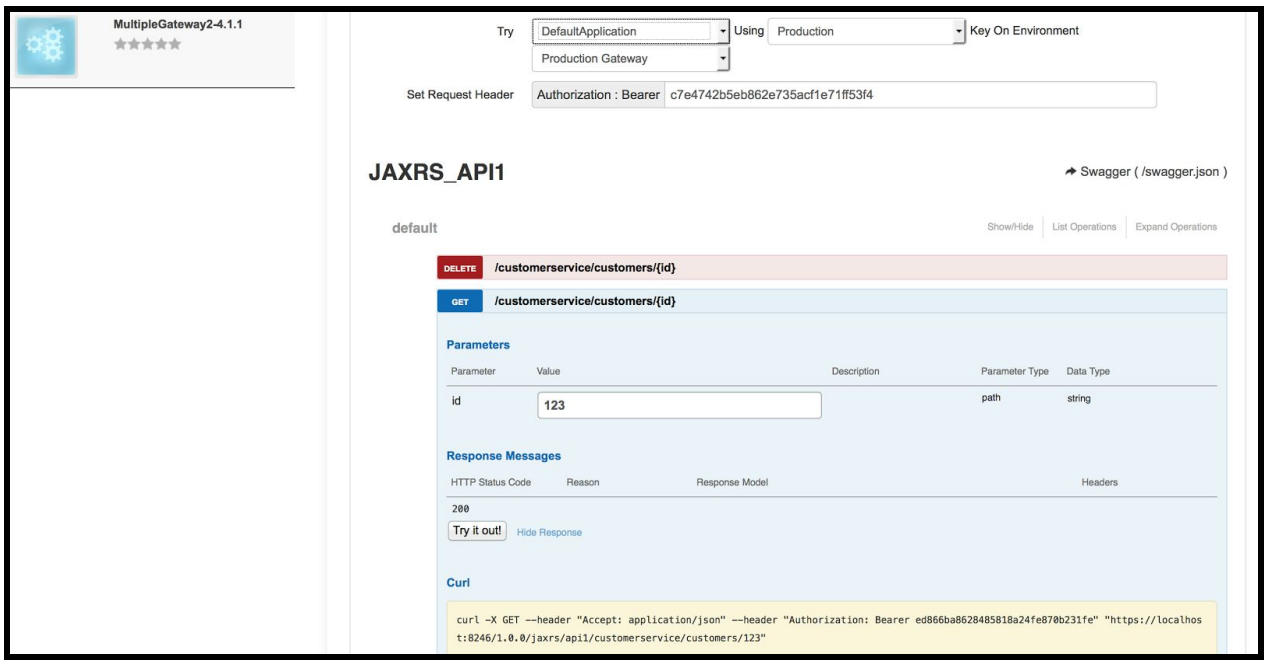
```
{
  "Customer": {
    "id": 124,
    "name": "Shiromalie"
  }
}
```

Response Code
200

Response Headers
{
"content-type": "application/json"
}

Step 3 - Invoke the GET resource

- For GET method you do not have to provide a post body. All you have to do is pass the ID which was returned in the POST invocation above, i.e.:-124



Screenshot 3

Response of GET

When you click on **Try It Now** button, you should receive a response as below.

Response Body

```
{
  "Customer": {
    "id": 124,
    "name": "Shiromalie"
  }
}
```

Response Code

200

Response Headers

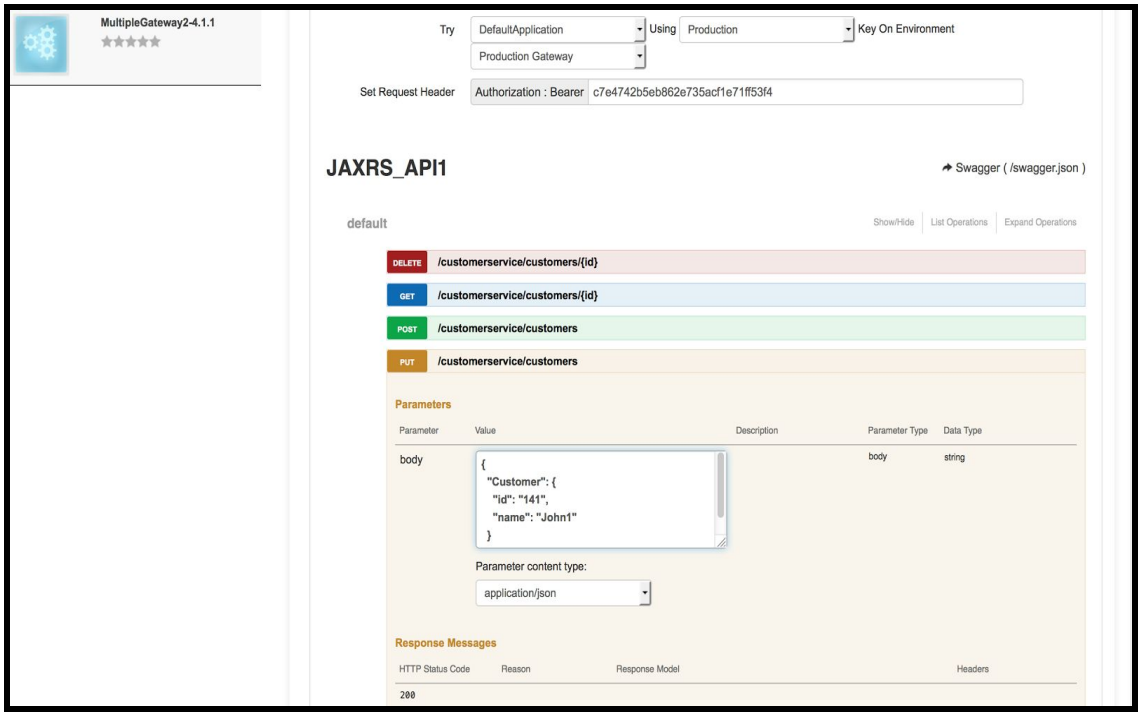
```
{

 "content-type": "application/json"

}
```

Step 4 **-** Invoke the PUT resource

```
{
  "Customer": {
    "id": "124",
    "name": "Evanthika"
  }
}
```



Screenshot 4

Response of PUT invocation

When you click on **Try It Now** button, you should receive a response as below.

Response Body no content **Response Code** 200

Step 4 **-** Invoke the GET resource once again

- As in Step 2 above, do another GET invocation to verify whether the PUT invocation has changed the customer name properly. If the PUT resource has been successful, the GET invocation response should be as below. Note the customer name highlighted in blue.

Response Body

```
{
  "Customer": {
    "id": 124,
    "name": "Evanthika"
  }
}
```

Response Code

200

Response Headers

```
{
 "content-type": "application/json"
}
```

Step 5 - Invoke the DELETE resource

- For DELETE method you do not have to provide a post body. All you have to do is pass the ID which was returned in the POST invocation above, i.e.:-124

Response of DELETE invocation

When you click on **Try It Now** button, you should receive a response as below.

Response Body
no content

Response Code
200

Response Headers
{
"content-type": "application/octet-stream"
}

Step 6 - Invoke the GET resource again to verify the success of the DELETE invocation

- As in Step 2 above, do another GET invocation to verify whether the DELETE invocation has deleted the customer details as expected. If the DELETE resource has been successful, the GET invocation response should be as below. Note the Response Code marked in blue.

Response Body
no content

Response Code
204

Response Headers
{
"content-type": null
}

Step 7 - Invoke the HEAD resource

- The HEAD method is used to verify whether the relevant resource exists in the service. Assume you are trying to verify the existence of the resource /customerservice/customers. Since you have given the URL pattern as /customerservice/customers in HEAD method while creating the API, you have to pass only the customer ID.

Response of HEAD invocation

When you click on **Try It Now** button, you should receive a response as below.

Response Body
no content

Response Code
200

Response Headers
{
"content-type": null
}