

## Hands-On Lab: Publishing SAGTours APIs to Developer Portal

### Objectives

In this hands-on lab, you will first group all three SAGTours REST APIs in API Gateway. Next, acting as the API Gateway Administrator Andy, you will publish the APIs to Developer Portal to expose them to external consumers. Eventually you will test them by using the Try API functionality in Developer Portal.

### Steps

- 1) Open the **Windows Services** panel and double-check that the following services, needed for API Gateway (two instances), the native services, and Developer Portal, are up and running. If a service is not running, start the service.
  - a) **Software AG Integration Server 10.11 (default)**
  - b) **Software AG Threat Protection Integration Server 10.11**
  - c) **Software AG Runtime 10.11**
- 2) Login to the (internal) **API Gateway** as user **Andy | manage**. Navigate to **APIs** and select the API **BookingsAPI**. Switch over to **Edit** mode and confirm with **Yes**.
  - a) Navigate to **API details > Basic information**.  
Adjust the following properties:
    - i) Maturity state: **Production**
    - ii) API grouping: *< from the API grouping dropdown values select **Product Catalog** >*

The screenshot shows the 'BookingsAPI' configuration page in the API Gateway. The 'Basic information' tab is active. The 'Name' field is 'BookingsAPI' and the 'Version' is '3.0'. The 'Maturity state' dropdown is set to 'Production' and the 'API grouping' dropdown is set to 'Product Catalog'. Both dropdowns are highlighted with red boxes. The 'API grouping' dropdown is also labeled 'Product Catalog' in the 'Action' column.

- b) To provide a sample request body, navigate to **API details** > **Resources and methods** and select the **/bookings** resource. Scroll down to the **POST** method. Click **Add request +**.

The screenshot shows the 'BookingsAPI' configuration page in the 'Resources and methods' section. The 'POST' method is selected. The description is 'Create a new booking for the customer. Input must be in JSON and the result when successful will be a JSON response with the booking.' The operation ID is 'POST\_BOOKING'. There is a table for method parameters with one entry: 'Authorization' with a description 'Basic Auth is needed to login for booking', type 'Header', data type 'String', required 'No', and repeat 'No'. At the bottom, there is a red box around the 'Add request +' button.

- c) Provide the following properties:
- i) New request: **< selected >**
  - ii) Content type: **application/json**
  - iii) Tab **Sample**: **< copy and paste the request body Sample for method POST from file BookingsAPI.txt as available in folder C:\Training\E456B-7BE\Lab15 >**

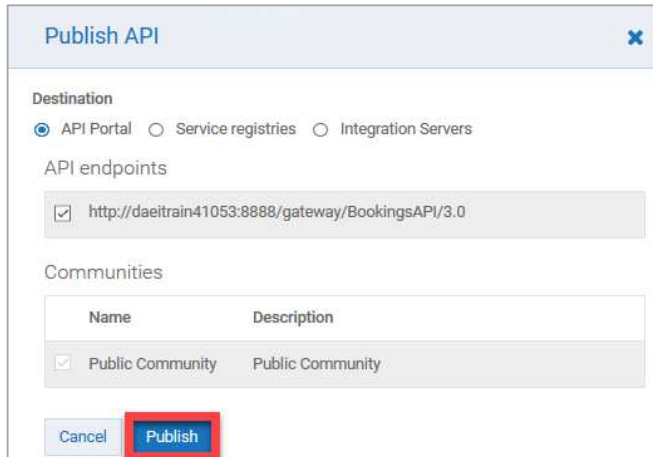
The screenshot shows the 'Add request' dialog in the 'BookingsAPI' configuration page. The 'New request' radio button is selected. The 'Content type' is set to 'application/json'. The 'Sample' tab is active, showing a JSON request body. The JSON body is: 

```
{  "roomID": "SBS",  "numRooms": "1",  "person": {    "emailAddress": "Adam.Apple@email.com",    "title": "Mr.",    "firstName": "Adam",    "surname": "Apple",    "dob": "1992.09.01",    "passportID": "1220123456789",    "passportExpiration": "2022.07.01",    "passportOrigin": "USA"  }  }
```

Click on **Add** next to Content type.

- d) Click **Save**.

- 3) In the (internal) API Gateway header menu, click **APIs**. Click the cloud icon with an upper arrow to publish API **BookingsAPI** to Developer Portal. Within the next dialog keep the default settings and click **Publish**.

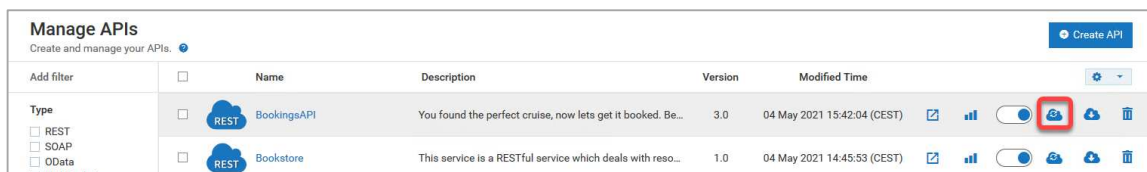








The 'Publish API' dialog box shows the following configuration:

- Destination:** ☒ API Portal, ☐ Service registries, ☐ Integration Servers
- API endpoints:** ☒ http://daeitrain41053:8888/gateway/BookingsAPI/3.0
- Communities:** ☒ Public Community, ☐ Public Community

Buttons at the bottom: Cancel, Publish (highlighted with a red box).

You will get a success message and the publish icon is replaced by a republish icon.



Manage APIs							Create API
Add filter		Name	Description	Version	Modified Time		
<b>Type</b> <input type="checkbox"/> REST <input type="checkbox"/> SOAP <input type="checkbox"/> OData	<input type="checkbox"/>	 BookingsAPI	You found the perfect cruise, now lets get it booked. Be...	3.0	04 May 2021 15:42:04 (CEST)		
	<input type="checkbox"/>	 Bookstore	This service is a RESTful service which deals with reso...	1.0	04 May 2021 14:45:53 (CEST)		

- 4) Open API **SearchCruise** in **Edit** mode.
- a) Navigate to **API details > Basic information** and adjust the following properties:
- i) Maturity state: **Production**
  - ii) API Grouping: **Search**
- b) **Save** your changes.

- 5) **Publish** the **SearchCruise** API to Developer Portal using the default settings.



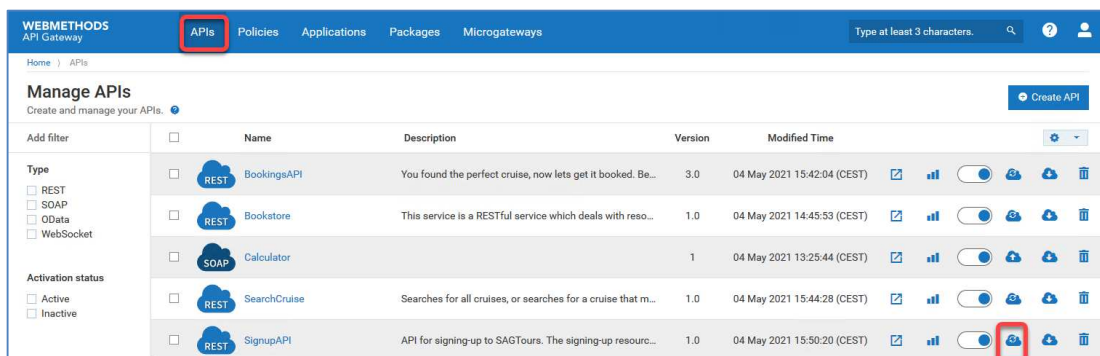
SearchCruise  
View API details, basic and technical information, resources and methods available, and API specifications: ...

Buttons:  Publish,  Edit,  Deactivate, 

Navigation tabs: API details, Scopes, Policies, Mashups, Applications, Analytics

- 6) Open API **SignupAPI** in **Edit** mode.
- a) Navigate to **API details > Basic information** and adjust the following properties:
- i) Maturity state: **Production**
  - ii) API grouping: **Customer Management**

- b) Navigate to **API details > Resources and methods** and select resource **/customer**.  
Scroll down to the **POST** method. Click **Add request +**.
  - c) Provide the following properties:
    - i) New request: **< selected >**
    - ii) Content type: **application/json**
    - iii) Tab **Sample**: **< copy and paste the request body Sample for method POST from file SignupAPI.txt as available in folder C:\Training\E456B-7BE\Lab15 >**
  - d) Click on **Add** next to Content type.
  - e) Navigate to **API Details > Resources and methods** and select resource **/customer/{customerID}**.  
Scroll down to the **PUT** method. Click **Add request +**.
  - f) Provide the following properties:
    - i) New request: **< selected >**
    - ii) Content type: **application/json**
    - iii) Tab **Sample**: **< copy and paste the request body sample for method PUT out of file SignupAPI.txt as available in folder C:\Training\E456B-7BE\Lab15 >**
  - g) Click on **Add** next to Content type.
  - h) **Save** your changes.
- 7) Still on the API details page, **publish** the **SignupAPI** API to Developer Portal. Use the default settings.
- 8) Click **APIs** in the (internal) API Gateway header menu to verify its state.

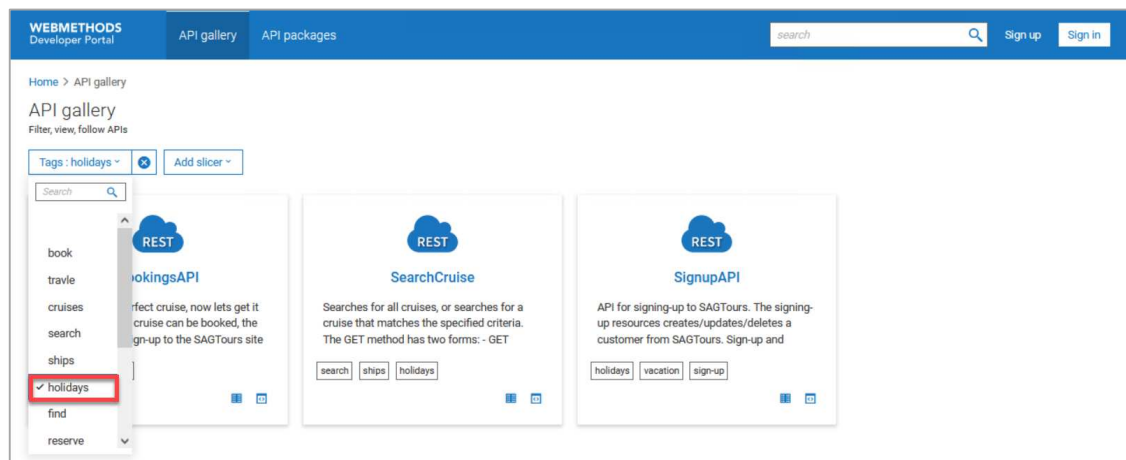


- 9) In your **Mozilla Firefox** browser, open a new window as **New Private Window** (Ctrl+Shift+P).

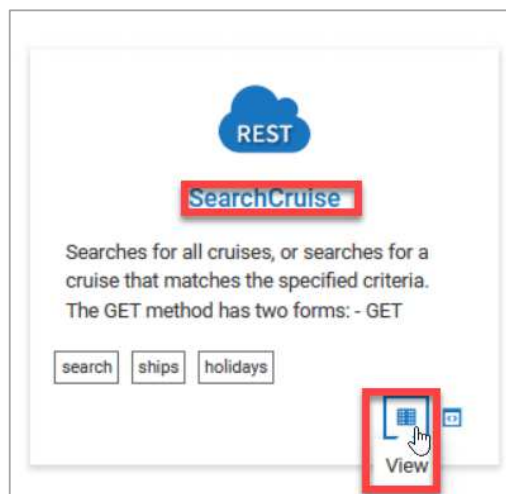
*Note:* This helps to avoid caching problems.

- 10) Use the new private window to connect to the **Developer Portal** using the URL <http://localhost:28101/portal> or by using the provided bookmark labeled as **Developer Portal**.

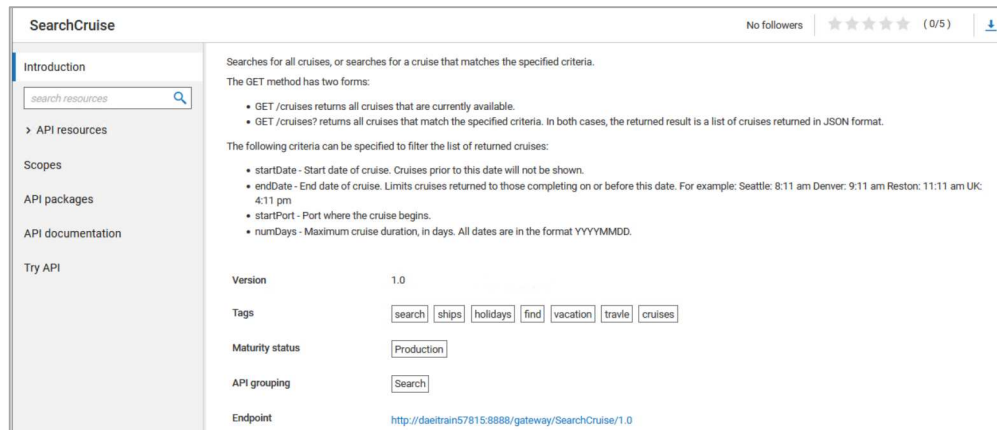
- a) To act like a guest user, do not sign-in. Instantly click the **API gallery** link in the header to see the registered APIs. By default, all APIs are listed. For filtering, add a slicer of type **Tags** and filter for all APIs tagged with tag **holidays**.



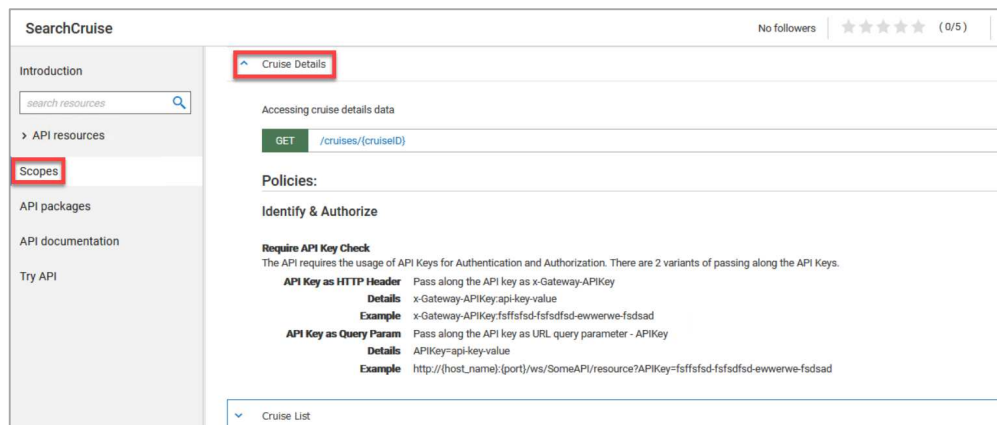
- b) From the list of “holidays” APIs open the details view of the **SearchCruise** API. To do so, click its name or use the view icon.



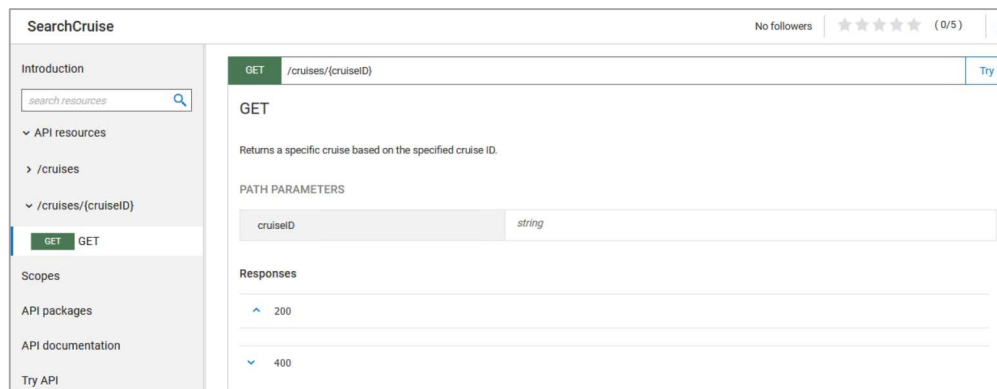
- c) On the left-hand side of the API's details view you find a navigation bar, listing the different sections of the API definition/description. By default, section Introduction is preselected. In the center you find the API description, version, tags, maturity status, endpoint, etc.



- d) Select **Scopes** from the left-hand navigation. Developer Portal will display the name of the API scopes you defined earlier in API Gateway: **Cruise List** and **Cruise Details**. Explore the scope definition, assigned resources and methods, and the applied policies of scope **Cruise Details**.



- e) Thereafter click on the **GET** method of resource **/cruises/{cruiseID}** to explore its parameters and defined responses.



11) In Developer Portal, get back to the **API gallery**.

- a) Open the details view of the **BookingsAPI**.
- b) Expand **API resources** in the left-hand navigation. All available REST resources and their supported methods as defined in API Gateway are shown. Click on each method to explore its details.

The screenshot shows the 'BookingsAPI' details page. On the left, the 'API resources' section is expanded, showing the 'POST /bookings' resource. The main content area displays the details for this resource, including a description, header parameters (Authorization), and a sample request body.

**POST /bookings**

Create a new booking for the customer. Input must be in JSON and the result when successful will be a JSON response with the booking.

**HEADER PARAMETERS**

Authorization: string  
Basic Auth is needed to login for booking

**REQUEST BODY**

No schema found

**Samples**

```
{  "cruiseID": "00001",  "customerID": "Adam.Apple@email.com",  "room": {    "roomID": "SDS",    "numRooms": "1"  },  "person": {    "emailAddress": "Adam.Apple@email.com",    "title": "Mr.",    "firstname": "Adam",  }}
```

- c) Before we test the BookingsAPI from Developer Portal, navigate to the description text shown in the **Introduction** section.  
The API developer wrote in this description text that the customer needs to sign-up to the SAG Tours site before he or she can book a cruise using the BookingsAPI. This sign-up must be done by calling the **SignupAPI** in advance.

The screenshot shows the 'BookingsAPI' details page, specifically the 'Introduction' section. The text describes the requirement for customers to sign-up to the SAG Tours site before using the BookingsAPI.

**BookingsAPI**

No followers | ★★★★★ (0/5) | Download

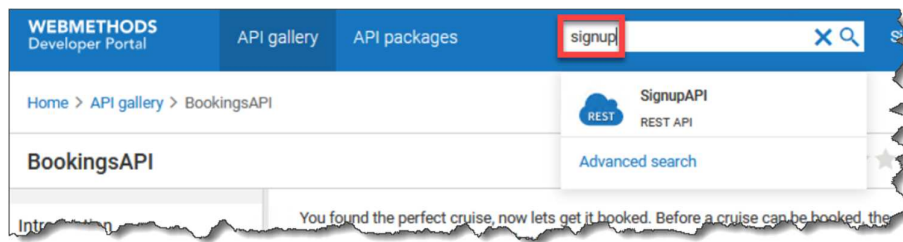
**Introduction**

search resources

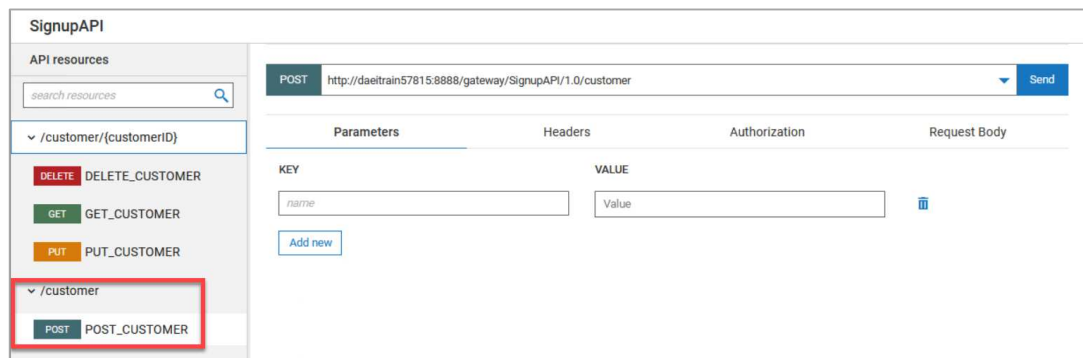
> API resources

You found the perfect cruise, now lets get it booked. Before a cruise can be booked, the customer must sign-up to the SAGTours site if they have not already done so. Once signed-up, use the Bookings API to book a cruise. You need the credentials you gave during sign-up to book the cruise. The credentials should be sent to the API using HTTP Basic Authentication (the HTTP header Authorization: BASIC ).

12) Try the search functionality in the Developer Portal header to search for the API **SignupAPI**.



- Open the API **SignupAPI** from the search result list.
- Select **Try API** from its left-hand list of links.
- Select the REST resource **/customer** and method **POST**.



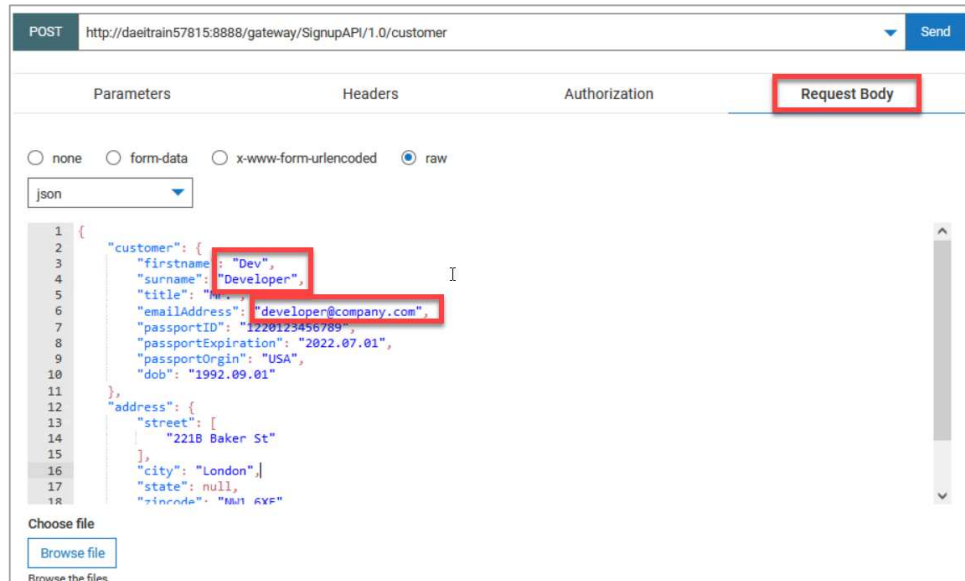
- On the **Headers** tab make sure that key-value pairs for **content-type** and for **accept** are set properly:
  - Content-type:** **application/json**
  - Accept:** **application/json**



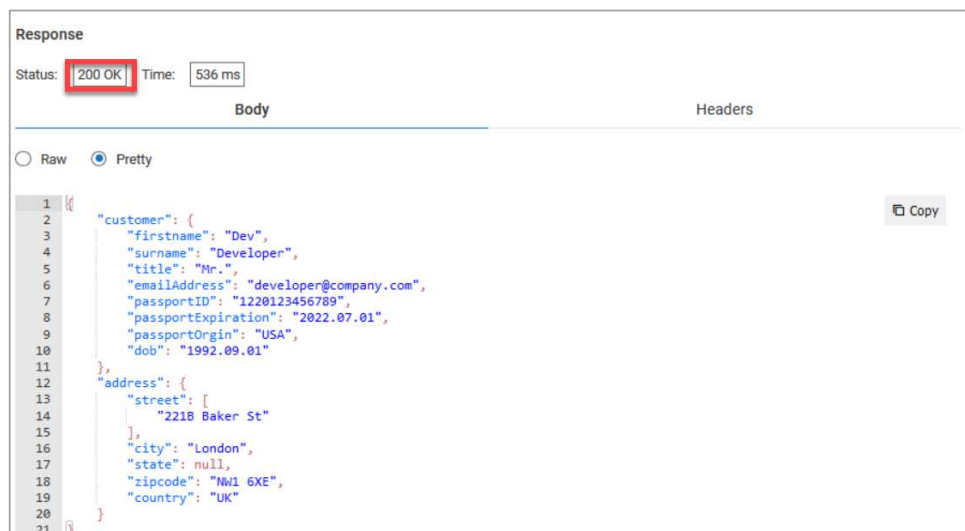


- e) On the **Request Body** tab replace the sample values in the request payload with the following customer data:
- i) firstname: **Dev**
  - ii) surname: **Developer**
  - iii) emailAddress: **developer@company.com**

Leave all the other values unchanged.



- f) Click the **Send** button.
- g) Review the response and verify that the response status is **200**.



**Note:** By this invocation, a new user/customer has been added to the SAGTours application. The provided email address will be persisted as the new customerID of the SAGTours customer.

13) Let's verify that a new customer has been added correctly to SAGTours:

- Still in the Try out panel of the **SignupAPI**, select the REST resource **/customer/{customerID}** from the left-hand navigation bar and select method **GET**.
- On the **Parameters** tab add the following request parameter:

**customerID**                      **developer@company.com**

SignupAPI

API resources

search resources

✓ /customer/{customerID}

DELETE DELETE\_CUSTOMER

GET GET\_CUSTOMER

PUT PUT\_CUSTOMER

> /customer

GET http://daeitrain57815.8888/gateway/SignupAPI/1.0/customer/developer@company.com

Parameters Headers Authorization Request Body

KEY VALUE

customerID developer@company.com

Add new

- On the **Headers** tab provide the following request header:

**Accept:**                      **application/json**

SignupAPI

API resources

search resources

✓ /customer/{customerID}

DELETE DELETE\_CUSTOMER

GET GET\_CUSTOMER

PUT PUT\_CUSTOMER

> /customer

GET http://daeitrain57815.8888/gateway/SignupAPI/1.0/customer/developer@company.com

Parameters Headers Authorization Request Body

KEY VALUE

Accept application/json

Add new

- The implementation of the SAGTours SignupAPI requires basic authentication for this resource-method combination. On the **Authorization** tab, select Authorization type **Basic Auth**. Provide the following credentials:
  - Username: **developer@company.com**
  - Password: **manage** (won't be checked by the application)

Parameters Headers Authorization Request Body

Authorization type

Basic Auth

Username

developer@company.com

Password

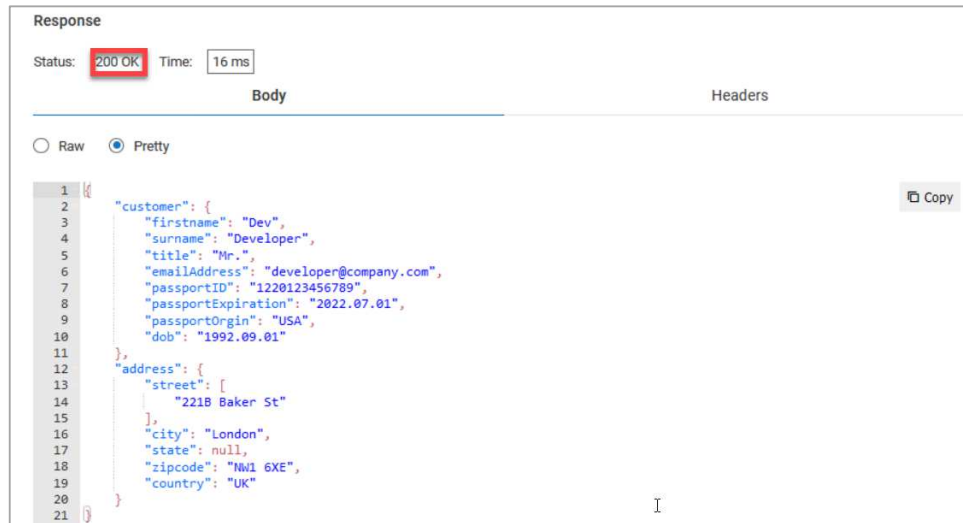
.....

Update

Click on **Update**.

- Click the **Send** button.

- f) Review the response and verify that the response Status is **200**.



*Note:* If you are getting a native service error with status code 400, close all Mozilla Firefox browser windows and connect to Developer Portal in a fresh browser window. Thereafter try it again.

- 14) Let's create a booking on behalf of the SAGTours user/customer just been added to SAGTours by using the SignupAPI.
- a) Open another private Window in Mozilla Firefox. Connect to **Developer Portal** using the pre-defined bookmark.
  - b) Act as an (unregistered) Developer Portal guest user, so do not sign-in.
  - c) Open the **BookingsAPI** from the **API gallery** and click the link **Tryout**.
  - d) Select the REST resource **/bookings** and method **POST**.
  - e) On the **Headers** tab configure the request headers:
    - i) **Accept:** **application/json**
    - ii) **Content-type:** **application/json**
  - f) On the **Request Body** tab replace the values in the request payload with the user data of our dummy customer created above:
    - i) emailAddress: **developer@company.com**
    - ii) firstname: **Dev**
    - iii) surname: **Developer**
    - iv) nameOnCreditCard **Dev Developer**

Leave all the other values unchanged.

- g) The booking service requires basic authentication. On the **Authorization** tab select Authorization type **Basic Auth**. Provide the following credentials:
- i) Username: **developer@company.com**
  - ii) Password: **manage** (won't be checked by the application):
- Click on **Update**.
- h) Click the **Send** button.
- i) Review the response and verify that the returned status is **200**.

The screenshot shows an API client interface with the following components:

- Left Panel:** A tree view showing the API structure. The selected path is `/bookings/{BookingID}`. Under `/bookings`, there are two methods: `POST POST_BOOKING` (highlighted) and `GET GET_LIST`.
- Parameters Tab:** Empty.
- Headers Tab:** Empty.
- Authorization Tab:** Active. It shows:
  - Authorization type:** A dropdown menu set to `Basic Auth`.
  - Username:** A text input field containing `developer@company.com`.
  - Password:** A text input field with masked characters (dots) and a visibility toggle icon.
  - Update:** A button to save the authorization settings.
- Response Section:**
  - Status:** `200 OK`
  - Time:** `51 ms`
  - Body:** A tab showing the response body in `Pretty` format. The JSON response is:

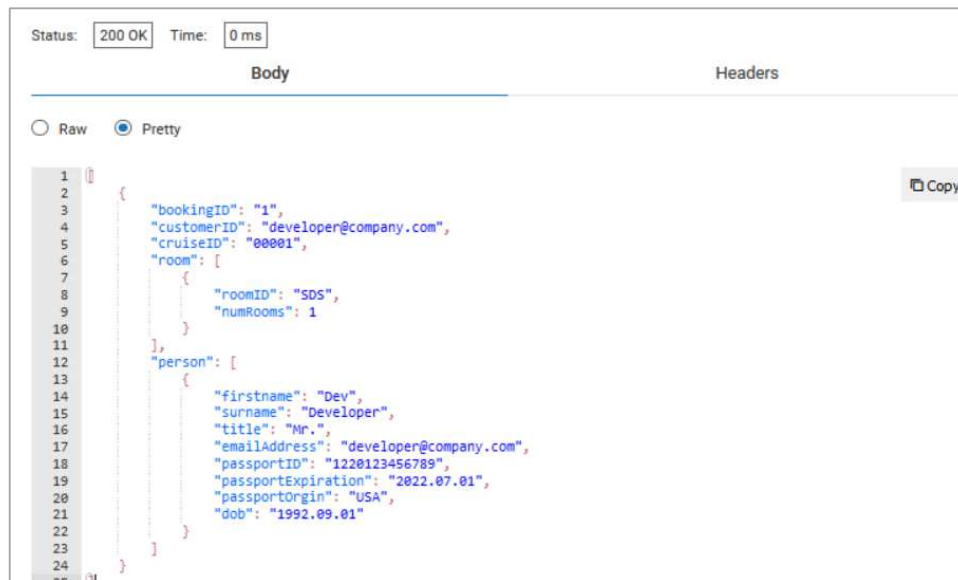
```
{
  "bookingID": "1",
  "customerID": "developer@company.com",
  "cruiseID": "00001",
  "room": {
    "roomID": "SDS",
    "numRooms": 1
  },
  "person": {
    "firstname": "Dev",
    "surname": "Developer",
    "title": "Mr.",
    "emailaddress": "developer@company.com",
    "passportID": "1220123456789",
    "passportExpiration": "2022.07.01",
    "passportOrigin": "USA",
    "dob": "1992.09.01"
  }
}
```

15) *For extra credit:*

Retrieve all the bookings made by the SAGTours user/customer you have added to SAGTours by using the SignupAPI.

- a) Open another private Window in Mozilla Firefox. Connect to **Developer Portal** using the pre-defined bookmark.
- b) Act as an (unregistered) API Gateway guest user, so do not sign-in.
- c) Open the **BookingsAPI** from the **API gallery** and click the link **Tryout**.
- d) Select the REST resource **/bookings** and method **GET**.

- e) On the **Headers** tab configure the request headers:
  - i) **Accept:** **application/json**
  - ii) **Content-type:** **application/json**
- f) The booking service still requires basic authentication. On the **Authorization** tab select Authorization type **Basic Auth**. Provide the following credentials:
  - i) Username: **developer@company.com**
  - ii) Password: **manage** (won't be checked by the application):Click on **Update**.
- g) Click the **Send** button.
- h) Review the response and verify that the returned status is **200**.



The screenshot shows a REST client interface with the following details:

- Status:** 200 OK
- Time:** 0 ms
- Body** tab is selected.
- Raw** and **Pretty** options are at the top left, with **Pretty** selected.
- Copy** button is at the top right.
- The response body is a JSON object with the following structure:

```
{  "bookingID": "1",  "customerID": "developer@company.com",  "cruiseID": "00001",  "room": [    {      "roomID": "SDS",      "numRooms": 1    }  ],  "person": [    {      "firstname": "Dev",      "surname": "Developer",      "title": "Mr.",      "emailAddress": "developer@company.com",      "passportID": "1220123456789",      "passportExpiration": "2022.07.01",      "passportOrigin": "USA",      "dob": "1992.09.01"    }  ]}
```