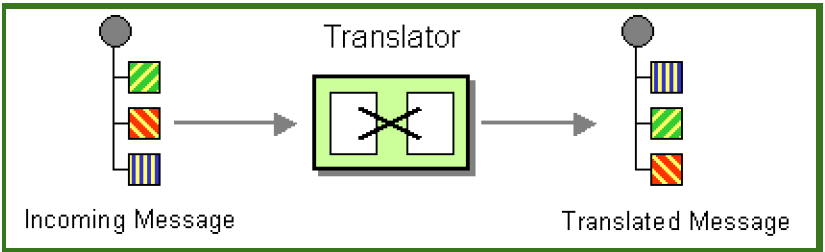
**Practical Exercise: Message Transformation**

Training Objective

Using WSO2 Integration Studio, practice configuring the core functionality of the product:

Message Transformation.

Enterprise Integration Pattern:Message Translator



High-level Steps

* Use the Data Mapper mediator.
* Redeploy and test the configuration.

Detailed Instructions

Translating Message Formats

What you'll build

Message transformation is necessary when the message format sent by the client is different from the message format expected by the back-end service. The **Message Translator** architectural pattern in WSO2 Micro Integrator describes how to translate from one data format to another.

**In this tutorial**, you send a request message to a back-end service where the format of the request payload is different to what is expected by the back-end service. The **Data Mapper** mediator is used to transform the request message payload to the format expected by the back-end service.

Let’s assume this is the format of the request sent by the client:

{

"name": "John Doe",

"dob": "1940-03-19",

"ssn": "234-23-525",

"address": "California",

"phone": "8770586755",

"email": "johndoe@gmail.com",

"doctor": "thomas collins",

"hospital\_id": "grandoaks",

"hospital": "grand oak community hospital",

"cardNo": "7844481124110331",

"appointment\_date": "2017-04-02"

}

However, the format of the message compatible with the back-end service is as follows:

{

"patient": {

"name": "John Doe",

"dob": "1990-03-19",

"ssn": "234-23-525",

"address": "California",

"phone": "8770586755",

"email": "johndoe@gmail.com",

"cardNo": "7844481124110331"

},

"doctor": "thomas collins",

"hospital\_id": "grandoaks",

"hospital": "grand oak community hospital",

"appointment\_date": "2017-04-02"

}

The client message format must be transformed to the back-end service message format within the In sequence.

Let's get started!

Step 1: Set up the workspace

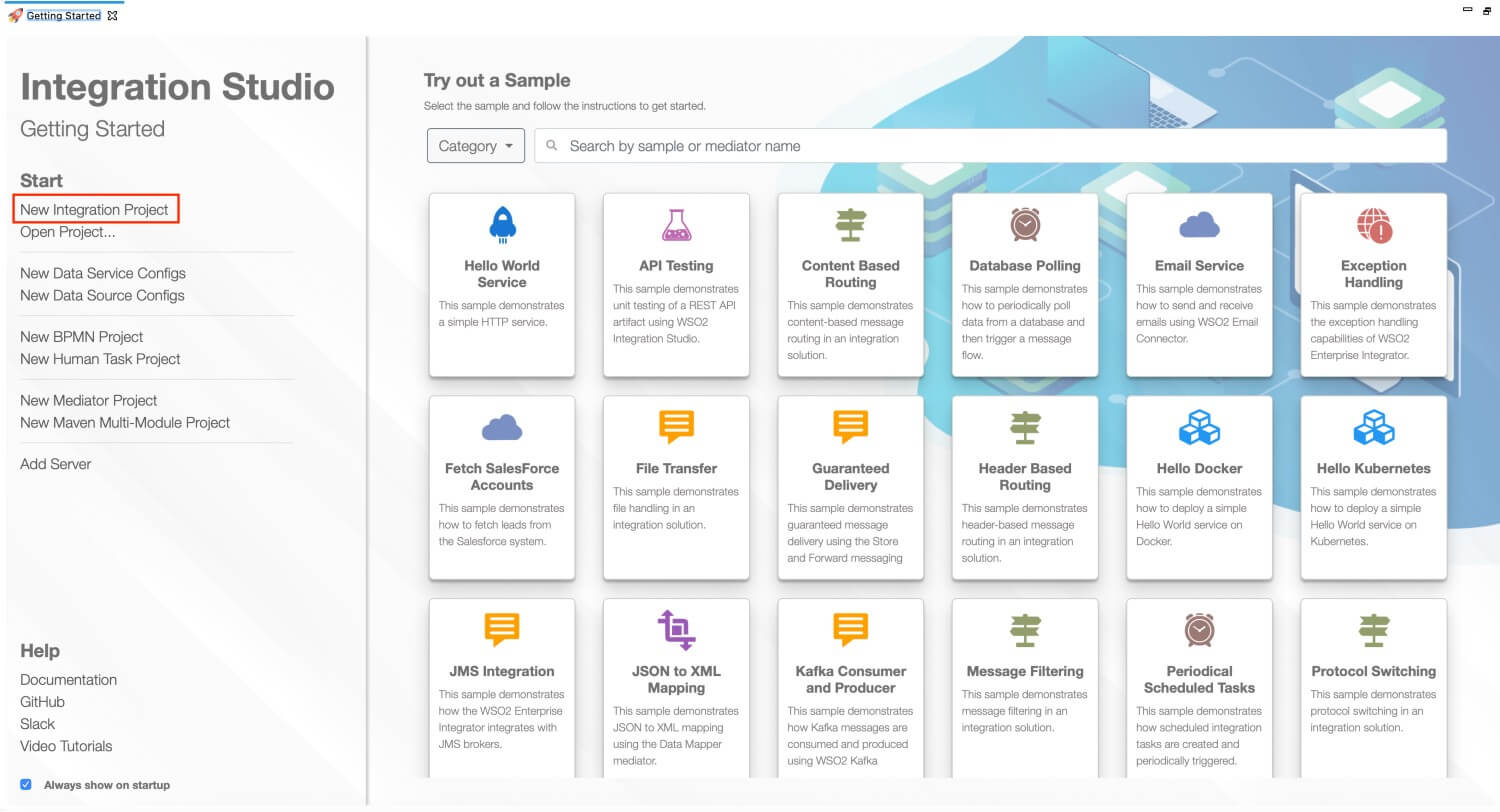
Download the relevant [WSO2 Integration Studio](https://wso2.com/api-management/tooling/) based on your operating system.

Step 2: Develop the integration artifacts

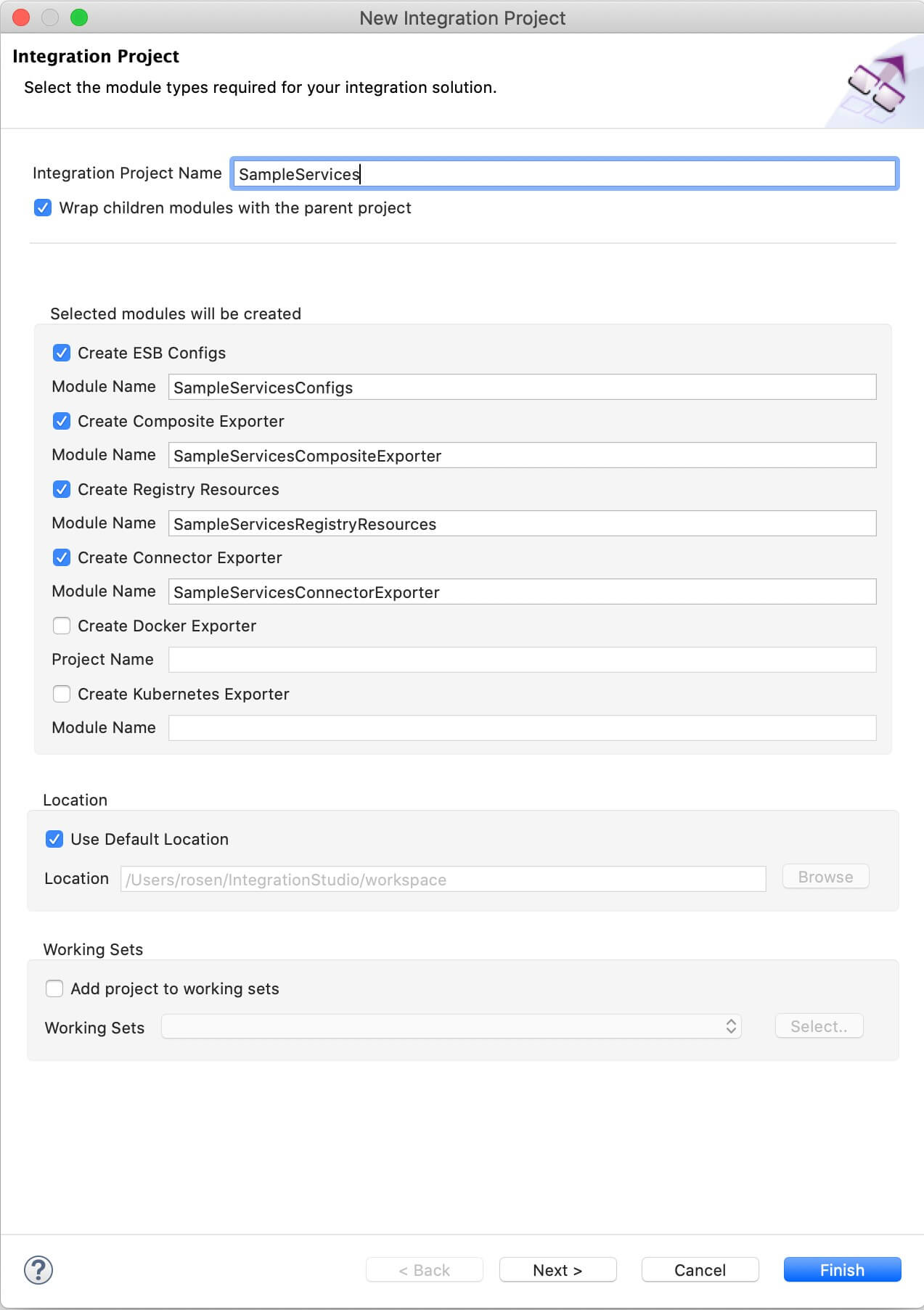
**Create an Integration project**

An Integration project is a maven multi module project, which will contain all the required modules for the integration solution.

1. Open **WSO2 Integration Studio**.
2. Click **New Integration Project** in the **Getting Started** tab as shown below.



This will open the **New Integration Project** dialog box.



1. Enter SampleServices as the project name and select the following check boxes to create the required modules.
   * **Create ESB Configs**
   * **Create Composite Exporter**
   * **Create Registry Resources**
2. Click **Finish**.

You will now see the projects listed in the **Project Explorer**.

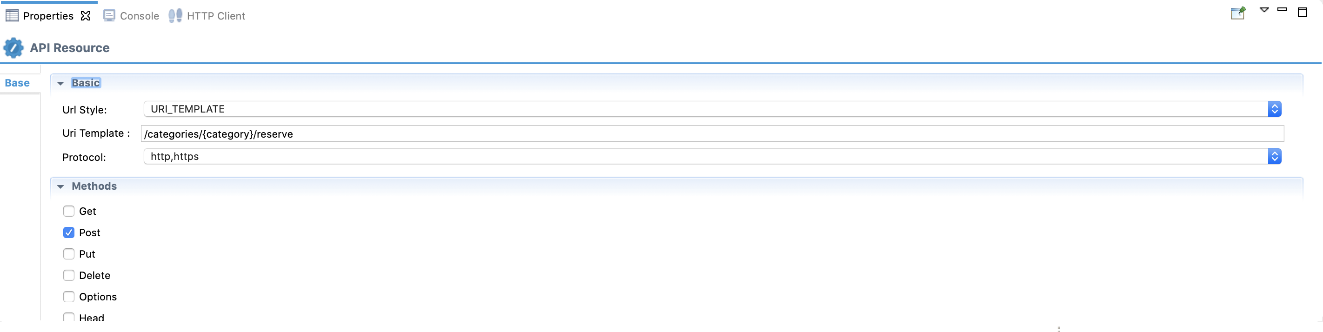
**Create a REST API**

1. In the Project Explorer, right-click **SampleServicesConfigs** and click **New -> REST API**.
2. Ensure **Create A New API Artifact** is selected and click **Next**.
3. Enter the details given below to create a new REST API.

|  |  |  |
| --- | --- | --- |
| Property | Value | Description |
| Name | HealthcareAPI | The name of the REST API. |
| Context | /healthcare | Here you are anchoring the API in the /healthcare context. This will become part of the name of the generated URL used by the client when sending requests to the Healthcare service. For example, setting the context to /healthcare means that the API will only handle HTTP requests where the URL path starts with http://host:port/healthcare. |
| Save location | SampleServicesConfigs | This is the **ESB Config** module where the artifact will be saved. |

1. Click the default API Resource to access the **Properties** tab and enter the following details:

|  |  |
| --- | --- |
| Property | Description |
| Url Style | Click in the **Value** field, click the down arrow, and select **URI\_TEMPLATE** from the list. |
| URI-Template | Enter /categories/{category}/reserve. |
| Methods | From the list of methods, select **POST**. |



**Create new Endpoint**

Let's create an Endpoint to represent the Hospital Service back-end service.

1. Right click **SampleServicesConfigs** in the project explorer and click **New -> Endpoint**.
2. Ensure **Create a New Endpoint** is selected and click **Next**.
3. Let's create the hospital service endpoint (**HospitalServicesEP**) using the following values:

|  |  |  |
| --- | --- | --- |
| Property | Value | Description |
| Endpoint Name | HospitalServicesEP | This is a single endpoint configured to forward requests to the relevant hospital by reading the hospital specified in the request payload. |
| Endpoint Type | HTTP Endpoint | Indicates that the back-end service is HTTP. |
| URI Template | http://localhost:9090/{uri.var.hospital}/categories/{uri.var.category}/reserve | The template for the request URL expected by the back-end service. The following two variables will be replaced by the corresponding values in the request message:   * + {uri.var.hospital}   + {uri.var.category} |
| Method | POST | Endpoint HTTP REST Method. |
| Save Endpoint in | SampleServicesConfigs | This is the ESB Config module we created in the last section. |

1. Click **Finish**.

**Define the mediation flow**

Let's configure the API resource with the data transformation logic.

1. Drag a **Property** mediator from the **Mediators** palette to the In Sequence of the API resource and name it **Get Hospital**.

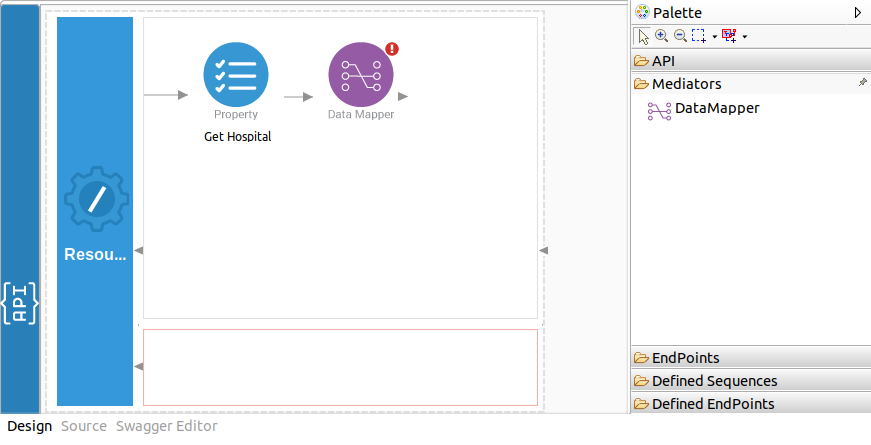
**Info**

This is used to extract the hospital name that is sent in the request payload.

1. With the **Property** mediator selected, access the **Properties** tab and give the following details:

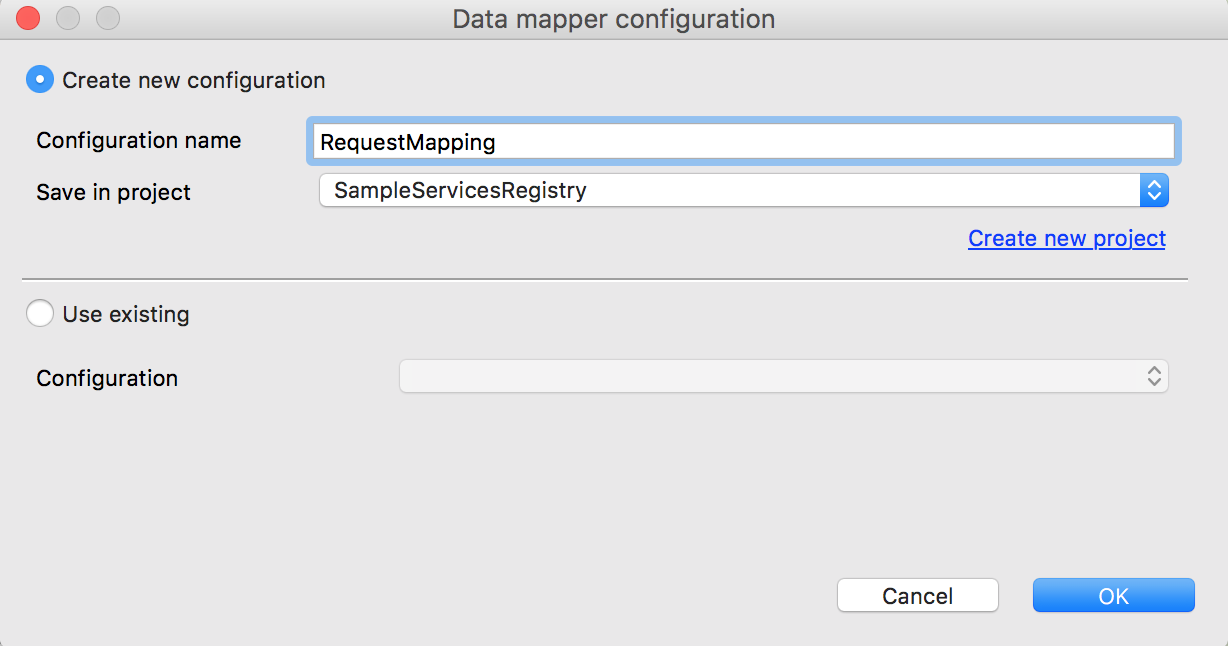
|  |  |  |
| --- | --- | --- |
| Property | Value | Description |
| Property Name | New Property... | Specifies that a new property is created. |
| New Property Name | uri.var.hospital | The name that will be used to refer this property's values. |
| Property Action | set | The property action. |
| Property Scope | default | The scope of the property. |
| Value (Expression) | json-eval($.hospital\_id) | Follow the steps given below to specify the expression value:  https://apim.docs.wso2.com/en/4.1.0/assets/img/integrate/tutorials/119132155/expression-value.png   * 1. Click the **Ex** button before the **Value** field. This specifies the value type as *expression*.   2. Now, click the **f** button to open the **Expression Selector** dialog box.   3. Enter json-eval($.hospital\_id) as the expression value.   **Note**: This is the JSONPath expression that will extract the hospital from the request payload. |

1. Add a **Data Mapper** mediator just after the Property mediator in the In Sequence of the API resource.

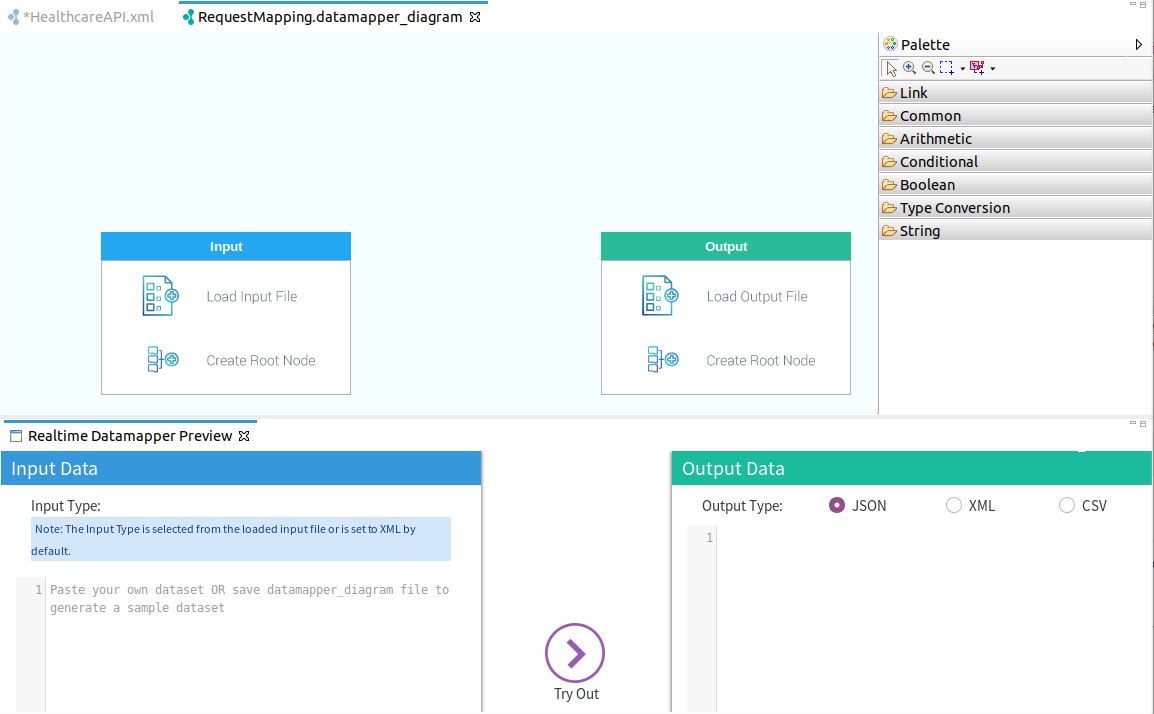


1. Double-click the Data Mapper mediator icon and specify the following details:

|  |  |
| --- | --- |
| Property | Description |
| Configuration Name | Enter RequestMapping as the name. |
| Save in project | Specify the **Registry Resource module** where the data mapper configuration should be saved. The **SampleServicesRegistryResources** module created at the time of creating the integration project will selected by default. |



1. Click **OK**. You can view the data mapping editor.



1. Create a JSON file (e.g., input.json) by copying the following sample content of the request message sent to the API resource and save it in your local file system.

{ "name": "John Doe",

"dob": "1990-03-19",

"ssn": "234-23-525",

"address": "California",

"phone": "8770586755",

"email": "johndoe@gmail.com",

"doctor": "thomas collins",

"hospital\_id": "grandoaks",

"hospital": "grand oak community hospital",

"cardNo": "7844481124110331",

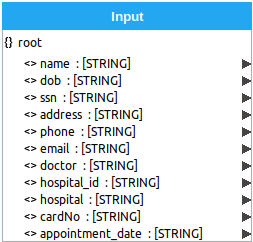
"appointment\_date": "2025-04-02"

}

**Info**

You can create a JSON schema manually for input and output using the **Data Mapper Diagram** editor.

1. Click **Load Input File** in the **Input** box to open the **Load Input** dialog box.
2. Select **JSON** as the **Resource Type**.
3. Click **file system** link in **Select resource from**, select the JSON file (i.e., input.json ) you saved in your local file system, and click **Open**. You can view the input format loaded in the **Input** box of the editor as shown below.



1. Create another JSON file (e.g., output.json) by copying the following sample content of the request message expected by the back-end service and save it in your local file system.

{

"patient": {

"name": "John Doe",

"dob": "1990-03-19",

"ssn": "234-23-525",

"address": "California",

"phone": "8770586755",

"email": "johndoe@gmail.com"

},

"doctor": "thomas collins",

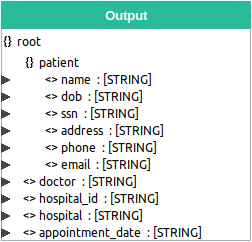
"hospital\_id": "grandoaks",

"hospital": "grand oak community hospital",

"appointment\_date": "2025-04-02"

}

1. Click **Load Output File** in the **Output** box to open the **Load Output** dialog box.
2. Select **JSON** as the **Resource Type**.
3. Click the **file system** link in **Select resource from**, select the JSON file you saved in your local file system, and click **Open**. You can view the input format loaded in the **Output** box in the editor as shown below.

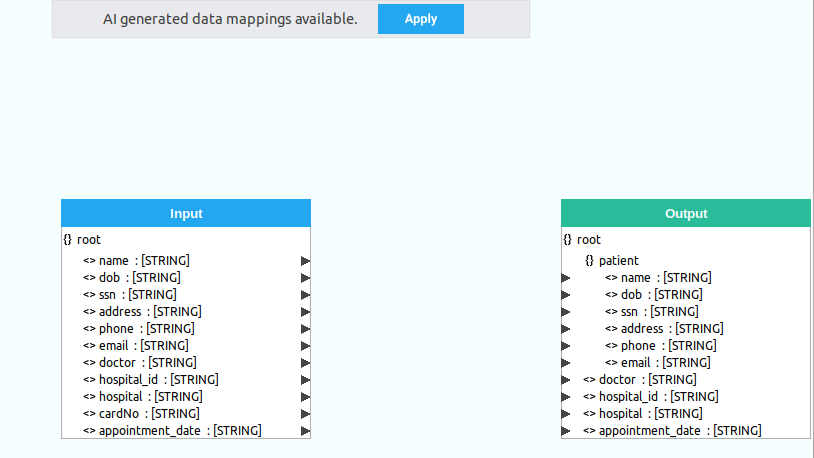


**Info**

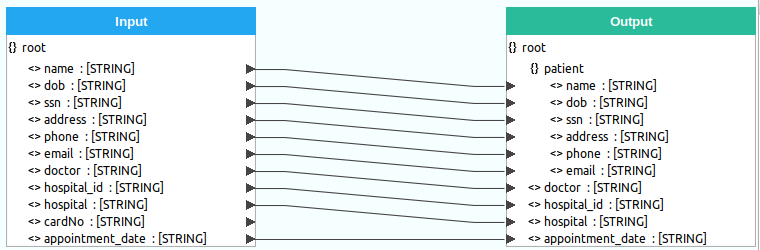
Check the **Input** and **Output** boxes with the sample messages to see if the element types (i.e. Arrays, Objects and Primitive values) are correctly identified. The following symbols will help you identify them correctly.

* 1. {} : represents object elements
  2. [] : represents array elements
  3. <> : represents primitive field values
  4. A : represents XML attribute value

1. Now, you need to map the input message with the output message. There are two ways to do the mapping:
   1. If you click **Apply**, the mapping will be generated by the **AI Data Mapper**. You have the option to manually change the mapping after it is generated.
   2. You can also manually draw the mapping by dragging arrows from the values in the **Input** box to the relevant values in the **Output** box.

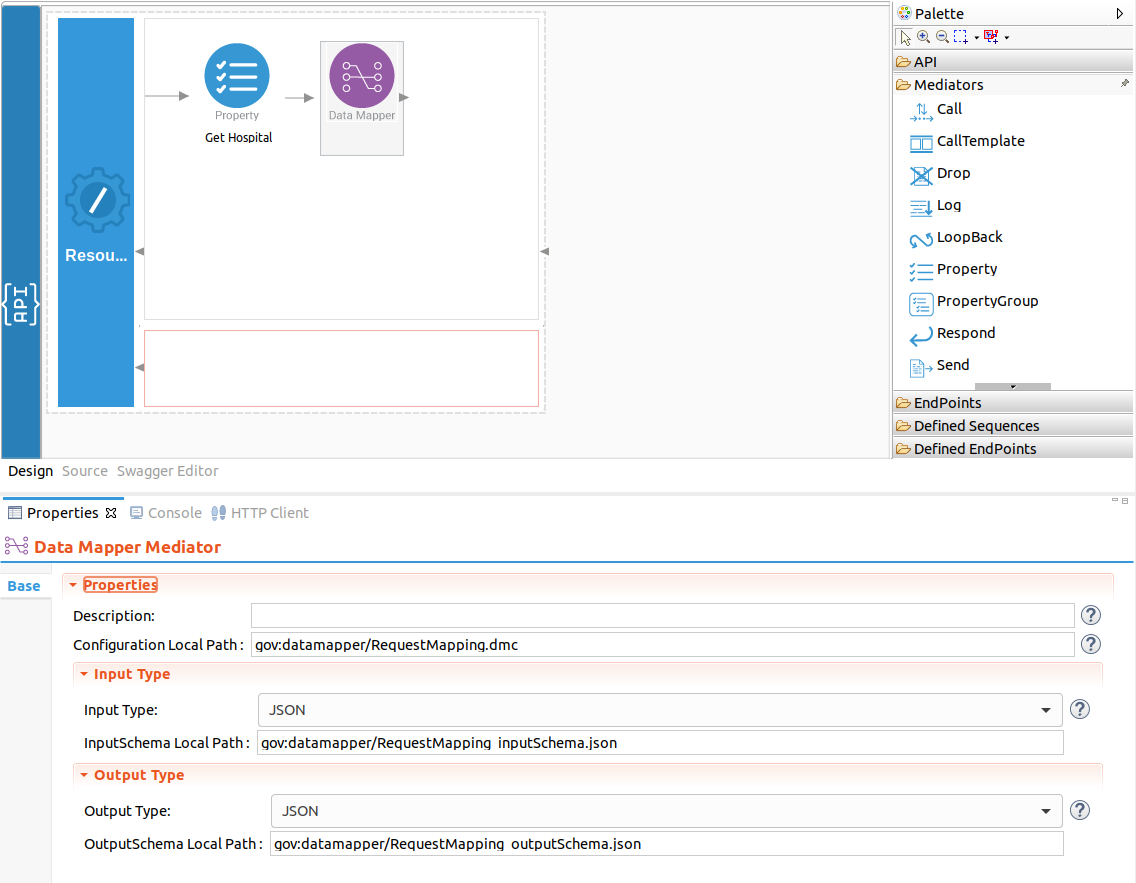


The completed mapping will look as follows:

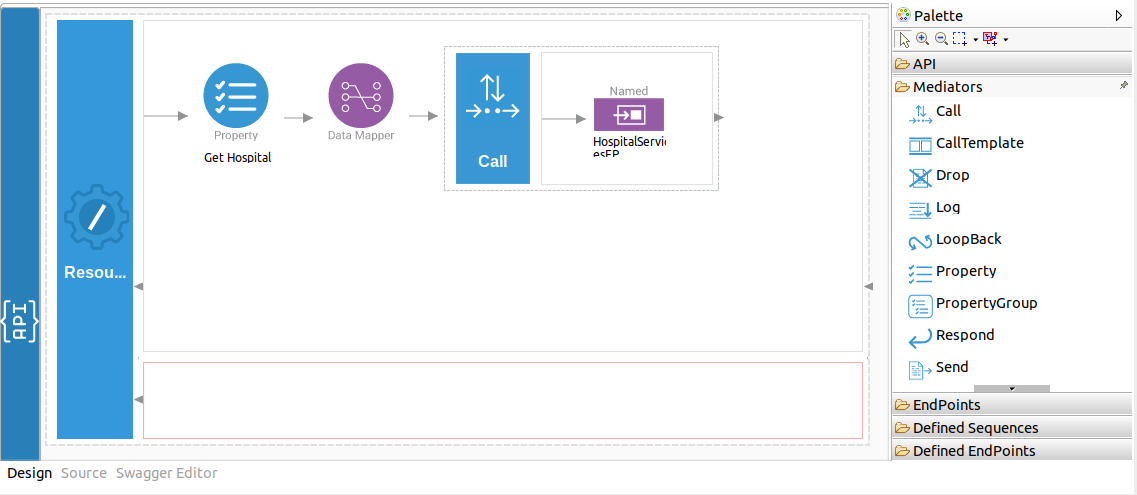


1. Save and close the configuration.
2. Go back to the **Design View** of the API Resource and select the **Data Mapper** mediator and edit the following in the **Properties** tab:

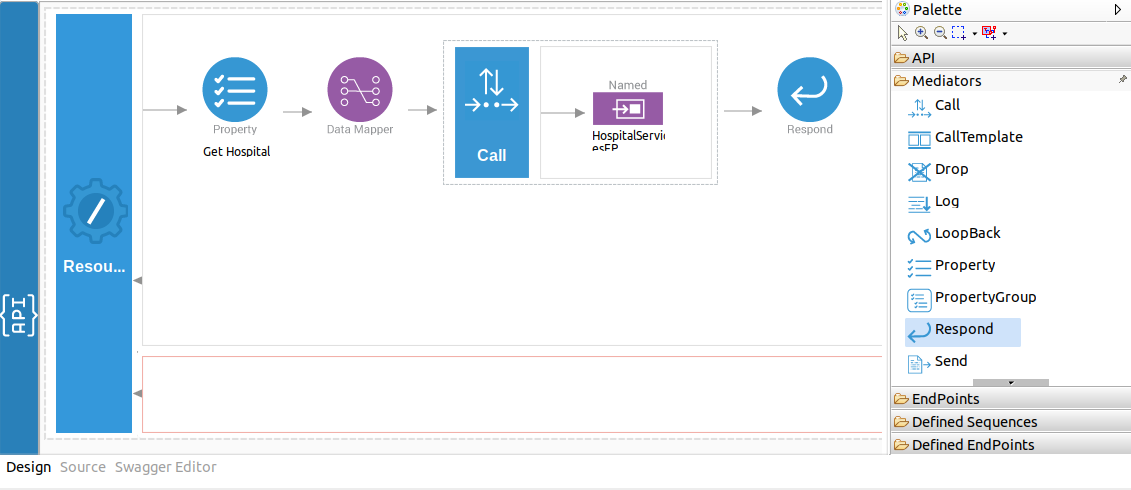
|  |  |
| --- | --- |
| Property | Description |
| Input Type | Select **JSON**. |
| Output Type | Select **JSON**. |



1. Add a Call mediator from the **Mediators** palette and add the HospitalServicesEP endpoint from the **Defined Endpoints** palette to the empty box adjoining the Call mediator.



1. Add a **Respond mediator** next to the **Call** mediator to return the response from the health care service back to the client.



1. Save the REST API configuration.

You have successfully created all the artifacts that are required for this use case.

Step 3: Package the artifacts

Package the artifacts in your composite exporter module (SampleServicesCompositeExporter) to be able to deploy the artifacts in the server.

1. Open the pom.xml file in the composite exporter module.
2. Ensure that the following projects and artifacts are selected in the POM file.
   * SampleServicesCompositeExporter
     + HealthcareAPI
     + HospitalServicesEP
   * SampleServicesRegistryResources
3. Save the changes.

Step 4: Build and run the artifacts

To test the artifacts, deploy the [packaged artifacts](https://apim.docs.wso2.com/en/latest/tutorials/integration-tutorials/transforming-message-content/#step-3-package-the-artifacts) in the embedded Micro Integrator:

1. Right-click the composite exporter module and click **Export Project Artifacts and Run**.
2. In the dialog box that opens, confirm that the required artifacts from the composite exporter module are selected.
3. Click **Finish**.

The artifacts will be deployed in the embedded Micro Integrator and the server will start.

* See the startup log in the **Console** tab.
* See the URLs of the deployed services and APIs in the **Runtime Services** tab.

Step 5: Test the use case

Let's test the use case by sending a simple client request that invokes the service.

**Start the back-end service**

1. Download the JAR file of the back-end service from [here](https://github.com/wso2-docs/WSO2_EI/blob/master/Back-End-Service/Hospital-Service-JDK11-2.0.0.jar).
2. Open a terminal, navigate to the location where your saved the back-end service.
3. Execute the following command to start the service:

java -jar Hospital-Service-JDK11-2.0.0.jar

**Send the client request**

Let's send a request to the API resource to make a reservation. You can use the embedded **HTTP Client** of WSO2 Integration Studio as follows:

1. Open the Postman application. If you do not have the application, download it from here : [Postman](https://www.postman.com/downloads/)
2. Add the request information as given below and click the **Send** button.

|  |  |
| --- | --- |
| Method | POST |
| Headers | Content-Type=application/json |
| URL | http://localhost:8290/healthcare/categories/surgery/reserve   * + The URI-Template format that is used in this URL was defined when creating the API resource: http://host:port/categories/{category}/reserve. |
| Body | { "name": "John Doe", "dob": "1990-03-19", "ssn": "234-23-525", "address": "California", "phone": "8770586755", "email": "johndoe@gmail.com", "doctor": "thomas collins", "hospital\_id": "grandoaks", "hospital": "grand oak community hospital", "cardNo": "7844481124110331", "appointment\_date": "2025-04-02" }     * + This JSON payload contains details of the appointment reservation, which includes patient details, doctor, hospital, and data of appointment. |

If you want to send the client request from your terminal:

1. Install and set up [cURL](https://curl.haxx.se/" \t "_blank) as your REST client.
2. Create a JSON file named request.json with the following request payload.

{

"name": "John Doe",

"dob": "1990-03-19",

"ssn": "234-23-525",

"address": "California",

"phone": "8770586755",

"email": "johndoe@gmail.com",

"doctor": "thomas collins",

"hospital\_id": "grandoaks",

"hospital": "grand oak community hospital",

"cardNo": "7844481124110331",

"appointment\_date": "2025-04-02"

}

1. Open a terminal and navigate to the directory where you have saved the request.json file.
2. Execute the following command.

curl -v -X POST --data @request.json http://localhost:8290/healthcare/categories/surgery/reserve --header "Content-Type:application/json"

**Analyze the response**

You will see the following response received to your **HTTP Client**:

{"appointmentNumber":1,

"doctor":

{"name":"thomas collins",

"hospital":"grand oak community hospital",

"category":"surgery","availability":"9.00 a.m - 11.00 a.m",

"fee":7000.0},

"patient":

{"name":"John Doe",

"dob":"1990-03-19",

"ssn":"234-23-525",

"address":"California",

"phone":"8770586755",

"email":"johndoe@gmail.com"},

"fee":7000.0,

"confirmed":false,

"appointmentDate":"2025-04-02"

}

You have now explored how the Micro Integrator can receive a message in one format and transform it into the format expected by the back-end service using the Data Mapper mediator.