Exposing several services as a single service

What you'll build[¶](https://apim.docs.wso2.com/en/latest/tutorials/integration-tutorials/exposing-several-services-as-a-single-service/#what-youll-build)

When information from several services are required for constructing a response to a client request, service chaining needs to be implemented. That is, several services are integrated based on some business logic and exposed as a single, aggregated service.

In this tutorial, when a client sends a request for a medical appointment, the Micro Integrator performs several service call to multiple back-end services in order to construct the response that includes all the necessary details. The **Call** mediator allows you to specify all service invocations one after the other within a single sequence.

You will also use the **PayloadFactory** mediator to take the response from one back-end service and change it to the format that is accepted by the other back-end service.

Concepts and artifacts used[¶](https://apim.docs.wso2.com/en/latest/tutorials/integration-tutorials/exposing-several-services-as-a-single-service/#concepts-and-artifacts-used)

* REST API
* HTTP Endpoint
* Property Mediator
* Call Mediator
* PayloadFactory Mediator

Graphical user interface, text, application

Description automatically generated

Let's get started![¶](https://apim.docs.wso2.com/en/latest/tutorials/integration-tutorials/exposing-several-services-as-a-single-service/#lets-get-started)

Step 1: Set up the workspace[¶](https://apim.docs.wso2.com/en/latest/tutorials/integration-tutorials/exposing-several-services-as-a-single-service/#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[¶](https://apim.docs.wso2.com/en/latest/tutorials/integration-tutorials/exposing-several-services-as-a-single-service/#step-2-develop-the-integration-artifacts)

**Create an Integration project**[**¶**](https://apim.docs.wso2.com/en/latest/tutorials/integration-tutorials/exposing-several-services-as-a-single-service/#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.

Calendar

Description automatically generated

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

Graphical user interface, application

Description automatically generated

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

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

**Create new Endpoints**[**¶**](https://apim.docs.wso2.com/en/latest/tutorials/integration-tutorials/exposing-several-services-as-a-single-service/#create-new-endpoints)

Let's create three HTTP endpoints to represent all three back-end services: Hospital Service, Channeling Service, Payment 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 |  |
| Endpoint Name | HospitalServicesEP |  |
| Endpoint Type | HTTP Endpoint |  |
| URI Template | http://localhost:9090/{uri.var.hospital}/categories/{uri.var.category}/reserve |  |
| Method | POST |  |
| Static Endpoint |  |  |
| Save Endpoint in | SampleServicesConfigs |  |

1. Click **Finish**.
2. Create another endpoint for the Channeling back-end service and specify the details given below:

|  |  |  |
| --- | --- | --- |
| Property | Value |  |
| Endpoint Name | ChannelingFeeEP |  |
| Endpoint Type | HTTP Endpoint |  |
| URI Template | http://localhost:9090/{uri.var.hospital}/categories/appointments/{uri.var.appointment\_id}/fee |  |
| Method | GET |  |
| Static Endpoint |  |  |
| Save Endpoint in | SampleServicesConfigs |  |

1. Click **Finish**.

|  |  |  |
| --- | --- | --- |
| Property | Value | Description |
| Endpoint Name | SettlePaymentEP | The name of the endpoint. |
| Endpoint Type | HTTP Endpoint | Indicates that the back-end service is HTTP. |
| URI Template | http://localhost:9090/healthcare/payments | The template for the request URL expected by the back-end service. |
| Method | POST | This endpoint artifact will be used to post informtion to the back-end service. |
| Static Endpoint |  | Select this option because we are going to use this endpoint only in this ESB Config module and will not reuse it in other projects.  **Note**: If you need to create a reusable endpoint, save it as a Dynamic Endpoint in either the Configuration or Governance Registry. |
| Save Endpoint in | SampleServicesConfigs | This is the ESB Config module. |

1. Create another endpoint for the Settle Payment back-end service and specify the details given below:
2. Click **Finish**.

You have now created the endpoints that are required for this tutorial.

**Create a REST API**[**¶**](https://apim.docs.wso2.com/en/latest/tutorials/integration-tutorials/exposing-several-services-as-a-single-service/#create-a-rest-api)

1. In the Project Explorer, right-click **SampleServicesConfigs** and navigate to **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 | Value | Description |
| Url Style | URI\_TEMPLATE | You can now specify dynamic variables to extract values from the request URL. |
| URI-Template | Enter /categories/{category}/reserve. | The request URL should match this template. The {category} variable will be replaced with the value sent in the request. |
| Methods | POST | This API resource will accept POST requests. |

**Update the mediation flow**[**¶**](https://apim.docs.wso2.com/en/latest/tutorials/integration-tutorials/exposing-several-services-as-a-single-service/#update-the-mediation-flow)

You can now start updating the API resource with the mediation flow.

1. Open the REST API resource. You will see the canvas for the in sequence and out sequence as shown below.
2. Drag a **Property** mediator from the **Mediators** palette to the In Sequence of the API resource and name it **Get Hospital**. This is used to extract the hospital name that is sent in the request payload.
3. With the **Property** mediator selected, access the **Properties** tab and give the following details:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Property | Value |  |  |  |
| Property Name | New Property... |  |  |  |
| New Property Name | uri.var.hospital |  |  |  |
| Property Action | set |  |  |  |
| Property Scope | default |  |  |  |
| Value (Expression) | json-eval($.hospital\_id) |  |  |  |

1. Add a new **Property** mediator just after the **Get Hospital** property mediator and name it **Get Card Number**. This will retrieve and store the card number that is sent in the request payload.
2. With the Property mediator selected, access the Properties tab and specify the following details:

|  |  |  |
| --- | --- | --- |
| Property | Value | Description |
| Property Name | New Property... | Specify a new property. |
| New Property Name | card\_number | The name of the property, which will be used to refer this property. |
| Property Action | set | The property action. |
| Value (Expression) | json-eval($.cardNo) | Follow the steps given below to specify the expression:     * 1. Click the **Ex** button before the **Value** field. This specifies the value type as *expression*.   2. Now, click the **f** button to open the **Expression Selector** dialog box.   3. Enter json-eval($.cardNo) as the expression value.   **Note**: This is the JSONPath expression that will extract the card number from the request payload. |
| Description | Get Card Number | The description of the property. |

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.

**Info**

Using the Call mediator allows us to define other service invocations following this mediator.

**Note**

The following response will be returend from GrandOakEP, ClemencyEP, or PineValleyEP:

{"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}

Let's use Property mediators to retrieve and store the values that you get from the response you receive from GrandOakEP, ClemencyEP, or PineValleyEP.

1. Add a Property mediator to retrieve and store the value sent as appointmentNumber.
2. With the Property mediator selected, access the Properties tab and specify the following details:

| Property |  | Value |
| --- | --- | --- |
| Property Name |  | **New Property** |
| New Property Name |  | uri.var.appointment\_id |
| Property Action |  | Select **set** |
| Value (Expression) |  | json-eval($.appointmentNumber) |
| Description |  | Get Appointment Number |

1. Similarly, add two more Property mediators. They will retrieve and store the doctor details and patient details respectively from the response that is received from GrandOakEP, ClemencyEP, or PineValleyEP.
   1. To store doctor details:

|  |  |  |
| --- | --- | --- |
| Property |  | Value |
| Property Name |  | **New Property** |
| New Property Name |  | doctor\_details |
| Property Action |  | **set** |
| Value (Expression) |  | json-eval($.doctor) |
| Description |  | Get Doctor Details |
|  |  |  |

* + To store patient details:

|  |  |
| --- | --- |
| Property | Description |
| Property Name | Select **New Property** |
| New Property Name | Enter patient\_details |
| Property Action | Select **set** |
| Value | json-eval($.patient) |  |

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

**Note**

The following response that is received from ChannelingFeeEP:

{"patientName":" John Doe ",

"doctorName":"thomas collins",

"actualFee":"7000.0"}

1. Add a Property mediator adjoining the Call mediator box to retrieve and store the value sent as actualFee.
2. Access the Property tab of the mediator and specify the following details:

|  |  |
| --- | --- |
| Property | Description |
| Property Name | Select **New Property** |
| New Property Name | Enter actual\_fee  **Note**: This value is used when invoking SettlePaymentEP. |
| Property Action | Select **set** |
| Value | Follow the steps given below to specify the expression:     * 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($.actualFee) as the expression value. |
| Description | Get Actual Fee |

1. Let's use the **PayloadFactory** mediator to construct the following message payload for the request sent to SettlePaymentEP.

{"appointmentNumber":2,

"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,

"card\_number":"1234567890"

}

1. Add a PayloadFactory mediator (from the **mediators** palette) next to the Property mediator to construct the above message payload.
2. With the PayloadFactory mediator selected, access the properties tab of the mediator and specify the following details:

| Property | Description |
| --- | --- |
| Payload Format | Select **Inline** |
| Media Type | Select **json** |
| Payload | {"appointmentNumber":$1, "doctor":$2, "patient":$3, "fee":$4, "confirmed":"false", "card\_number":"$5"}  This is the message payload to send with the request to SettlePaymentEP. In this payload, $1, $2, $3, $4, and $5 indicate variables. |

1. To add the arguments for the PayloadFactory mediator:
   1. Click the **plus** icon (A picture containing text, first-aid kit, picture frame

      Description automatically generated) in the **Args** field to open the **PayloadFactoryArgument** dialog.
   2. Enter the following information in the **PayloadFactoryArgument** dialog box. This provides the argument that defines the actual value of the first variable (used in the format definition given in the previous step).

|  |  |
| --- | --- |
| Property | Description |
| Argument Type | Select Expression. |
| Argument Expression | Follow the steps given below to specify the expression:   * + 1. Click the text box for the **Argument Expression** field. This opens the **Expression Selector** dialog.     2. Select **Expression** from the list.     3. Enter $ctx:uri.var.appointment\_id. Note that the $ctx method is similar to using the get-property method. This method checks in the message context.     4. Click **OK.** |
| Evaluator | Select xml.  This indicates that the expression is provided in XML. |

1. Similarly, click **Add** and add more arguments to define the other variables that are used in the message payload format definition. Use the following as the **Value** for each of them:
   1. $ctx:doctor\_details
   2. $ctx:patient\_details
   3. $ctx:actual\_fee
   4. $ctx:card\_number
2. Add a Call mediator and add SettlePaymentEP from the Defined Endpoints palette to the empty box adjoining the Call mediator.
3. Add a **Respond** mediator to send the response to the client.

Step 3: Package the artifacts[¶](https://apim.docs.wso2.com/en/latest/tutorials/integration-tutorials/exposing-several-services-as-a-single-service/#step-3-package-the-artifacts)

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

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

Step 4: Build and run the artifacts[¶](https://apim.docs.wso2.com/en/latest/tutorials/integration-tutorials/exposing-several-services-as-a-single-service/#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/exposing-several-services-as-a-single-service/#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[¶](https://apim.docs.wso2.com/en/latest/tutorials/integration-tutorials/exposing-several-services-as-a-single-service/#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**[**¶**](https://apim.docs.wso2.com/en/latest/tutorials/integration-tutorials/exposing-several-services-as-a-single-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**[**¶**](https://apim.docs.wso2.com/en/latest/tutorials/integration-tutorials/exposing-several-services-as-a-single-service/#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 **HTTP Client** of WSO2 Integration Studio.

**Tip**

If you don't see the **HTTP Client** pane, go to **Window -> Show View - Other** and select **HTTP Client** to enable the client pane.

1. Enter the request information as given below and click the **Send** icon (Icon

   Description automatically generated).

|  |  |
| --- | --- |
| 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://:/categories/{category}/reserve. |
| Body | { "patient": { "name": "John Doe", "dob": "1940-03-19", "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": "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/) as your REST client.
2. Create a JSON file named request.json with the following request payload.

{

"patient": {

"name": "John Doe",

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

"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": "2025-04-02"

}

3. Open a terminal and navigate to the directory where you have saved the request.json file.

1. 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**[**¶**](https://apim.docs.wso2.com/en/latest/tutorials/integration-tutorials/exposing-several-services-as-a-single-service/#analyze-the-response)

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

{

"patient":"John Doe",

"actualFee":7000.0,

"discount":20,

"discounted":5600.0,

"paymentID":"480fead2-e592-4791-941a-690ad1363802",

"status":"Settled"

}

You can also try using any of the following parameters in your request payload.

For hospital:

* clemency medical center
* pine valley community hospital

Doctor Names:

* thomas collins
* henry parker
* abner jones
* anne clement
* thomas kirk
* cailen cooper
* seth mears
* emeline fulton
* jared morris
* henry foster

You have now explored how the Micro Integrator can do service chaining using the Call mediator and transform message payloads from one format to another using the **PayloadFactory** mediator.