**Request and Response Store in DB**

The Request Data Should be stored in data base and the response for that request should be store in data base.

Write the SQL Tables for storing the Request and Response.

In my example Request table = customer, Response table = response.

# Create tables to store requests and response.

Customer = 'CREATE TABLE `customer` ( `id` int NOT NULL AUTO\_INCREMENT, `FirstName` varchar(45) NOT NULL, `LastName` varchar(45) DEFAULT NULL, `flag` int DEFAULT '0', `request\_time` timestamp NULL DEFAULT CURRENT\_TIMESTAMP, PRIMARY KEY (`id`), CONSTRAINT `no\_space\_constraint\_firstName` CHECK ((length(`firstName`) > 0))) .

Response = 'CREATE TABLE `response` ( `id` int NOT NULL AUTO\_INCREMENT, `Status\_Message` varchar(255) NOT NULL, `response\_time` timestamp NULL DEFAULT CURRENT\_TIMESTAMP, `Error\_Messages` text NOT NULL, PRIMARY KEY (`id`), CONSTRAINT `no\_Space\_Status\_Message` CHECK ((length(`Status\_Message`) > 0))).

# Create the DSS

Create a new Data Service in the Data Service Configs.

Note: To Create a Data Service Please follow the basic document to create a Data Service.

Give the Credentials for the Data Source and Click on Save.

A screenshot of a computer

Description automatically generated with medium confidence

Next give the Query Configurations.

Write down the Sql Query Statement and click on Save.

A screenshot of a computer

Description automatically generated with medium confidence

Go to the resource configurations and configure the resource configurations.

A screenshot of a computer

Description automatically generated with medium confidence

Click on Save and Save the resource.

Next Export the data Service into the server and run the Server.

A screenshot of a computer

Description automatically generated with medium confidence

There you will get the url for the Data Service.

A screenshot of a computer

Description automatically generated with low confidence

Test the URL in the SOAP UI.

A screenshot of a computer

Description automatically generated with medium confidence

Endpoint for DSS will be as is on the address bar and the “\_putinsertintocustomer” Is the Header Action attribute for the DSS.

Test the Service with the input fields and after Successful execution implement the endpoint in the API.

# Defining the Endpoint.

Right click on the endpoint folder and click on new Endpoint.

Give the Configurations for the Address Endpoint and select the Format as SOAP 1.1.

And Finally specify the URL address.

A screenshot of a computer

Description automatically generated with medium confidence

# Implementing API

Right click on the API folder and create a new Rest API.

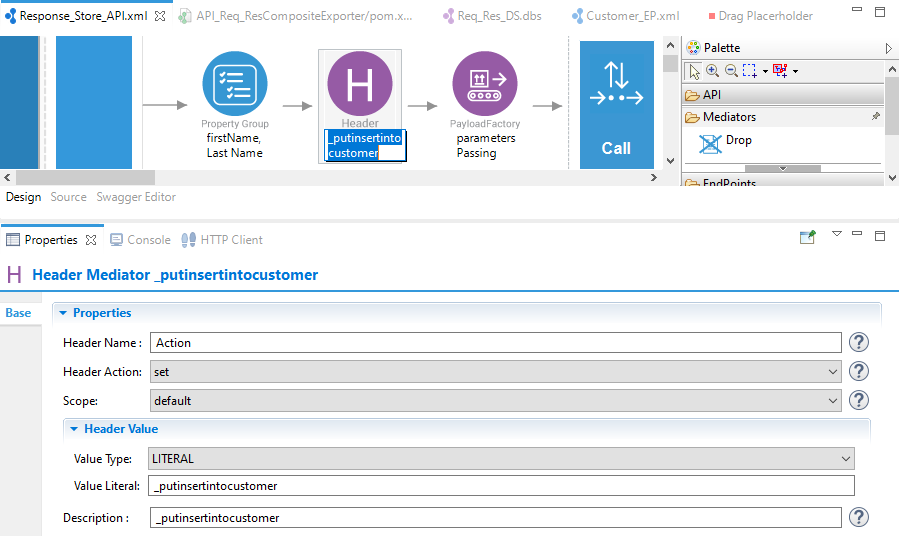
Select the API Resource and select the method as PUT.

Drag a property group mediator and give the properties which we want to take from the Request Body.

A screenshot of a computer

Description automatically generated with medium confidence

Place the Header Mediator and set the Configurations.



Set The Header Name as Action.

And give the literal value for the Action.

Drag the Payload Mediator and give the parameters whatever you want to pass to the Endpoint.

A screenshot of a computer

Description automatically generated

Drag a Call Mediator and place the defined Endpoint inside the Call Mediator.

A screenshot of a computer

Description automatically generated

After the Call Mediator Place the Log mediator to know the flow of work.

Next if we want our response payload in the JSON Format place a Property Mediator and set the Property name as messageType and value as application/json.

Next if we want to Store the Response Payload in a Property place a Property and Name that give the value as expression as json-eval($) to Store the Complete response.

Take a log mediator after the Response captured in the property and display it on the log.

Now Drag the DB Report Mediator and give the configurations for the DataBase.

A screenshot of a computer

Description automatically generated

Now write the sql Query to store the Data In the DataBase.

A screenshot of a computer

Description automatically generated

Click on finish.

Add the Respond Mediator at the end to get the Response.

In case of any failure of the inserting the data in the database the payload will go from the InSequence to the fault sequence, so we need to configure the api in the fault sequence in case of any error Failure.

Take a Property Mediator and give expression as json-eval($) in the expression and give any name to that property.

A screenshot of a computer

Description automatically generated with medium confidence

Drag a DB Report Mediator and give the credentials and write the sql statements to insert the Data.

A screenshot of a computer

Description automatically generated with medium confidence

Place the Log mediator next to DB Report and to see the flow of api.

Finally Add a Respond mediator.

# Sample End Point Configuration

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<endpoint name=*"Customer\_EP"* xmlns=*"http://ws.apache.org/ns/synapse"*>

<address format=*"soap11"* uri=*"http://KNOTPC38:8290/services/Req\_Res\_DS.SOAP11Endpoint"*>

<suspendOnFailure>

<initialDuration>-1</initialDuration>

<progressionFactor>1.0</progressionFactor>

</suspendOnFailure>

<markForSuspension>

<retriesBeforeSuspension>0</retriesBeforeSuspension>

</markForSuspension>

</address>

</endpoint>

# Sample API Source Code

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<api context=*"/responseStore"* name=*"Response\_Store\_API"* xmlns=*"http://ws.apache.org/ns/synapse"*>

<resource methods=*"PUT"*>

<inSequence>

<propertyGroup description=*"firstName, Last Name "*>

<property expression=*"json-eval($.FirstName)"* name=*"FirstName"* scope=*"default"* type=*"STRING"*/>

<property expression=*"json-eval($.LastName)"* name=*"LastName"* scope=*"default"* type=*"STRING"*/>

</propertyGroup>

<header description=*"\_putinsertintocustomer"* name=*"Action"* scope=*"default"* value=*"\_putinsertintocustomer"*/>

<payloadFactory description=*"parameters Passing"* media-type=*"json"*>

<format>{&#xd;

"FirstName" : "$1",&#xd;

"LastName" : "$2"&#xd;

}</format>

<args>

<arg evaluator=*"xml"* expression=*"$ctx:FirstName"*/>

<arg evaluator=*"xml"* expression=*"$ctx:LastName"*/>

</args>

</payloadFactory>

<call>

<endpoint key=*"Customer\_EP"*/>

</call>

<log description=*"Log After the Data Service Call"* level=*"full"*>

<property name=*"After Call"* value=*"After Call"*/>

<property expression=*"$body"* name=*"Payload"*/>

</log>

<property description=*"messageType"* name=*"messageType"* scope=*"axis2"* type=*"STRING"* value=*"application/json"*/>

<property description=*"REQUEST\_STATUS"* expression=*"json-eval($.REQUEST\_STATUS)"* name=*"Status"* scope=*"default"* type=*"STRING"*/>

<log description=*"After Getting Response"*>

<property expression=*"$ctx:Status"* name=*"REQUEST\_STATUS"*/>

</log>

<dbreport description=*"Response Table"*>

<connection>

<pool>

<driver>com.mysql.cj.jdbc.Driver</driver>

<url>jdbc:mysql://localhost:3306/wso2</url>

<user>root</user>

<password>mysql</password>

</pool>

</connection>

<statement>

<sql><![CDATA[insert into response (Status\_Message) values(?) ]]></sql>

<parameter expression=*"$ctx:Status"* type=*"VARCHAR"*/>

</statement>

</dbreport>

<respond/>

</inSequence>

<outSequence/>

<faultSequence>

<property description=*"err\_code"* expression=*"json-eval($)"* name=*"err\_code"* scope=*"default"* type=*"STRING"*/>

<dbreport description=*"Response Table"*>

<connection>

<pool>

<driver>com.mysql.cj.jdbc.Driver</driver>

<url>jdbc:mysql://localhost:3306/wso2</url>

<user>root</user>

<password>mysql</password>

</pool>

</connection>

<statement>

<sql><![CDATA[insert into response (Status\_Message,Error\_Messages) values(?,?) ]]></sql>

<parameter expression=*"$ctx:ERROR\_MESSAGE"* type=*"VARCHAR"*/>

<parameter expression=*"$ctx:err\_code"* type=*"VARCHAR"*/>

</statement>

</dbreport>

<log description=*"Log in Fault Sequence"* level=*"full"*>

<property expression=*"$ctx:ERROR\_CODE"* name=*"ERROR\_CODE"*/>

<property expression=*"$ctx:ERROR\_MESSAGE"* name=*"ERROR\_MESSAGE"*/>

<property expression=*"json-eval($)"* name=*"Error"*/>

</log>

<respond/>

</faultSequence>

</resource>

</api>