**ERROR\_HANDLING**

# ERRORS WHILE CALLING THE BACK-END SERVICES.

1. Incorrect Endpoint URL: One common error is providing an incorrect or invalid URL for the endpoint. Double-check the endpoint URL and ensure it is correctly formatted and accessible. Also, verify that the endpoint is up and running.
2. Connectivity Issues: Endpoint errors can be caused by network connectivity problems. Check the network configuration, firewall settings, and DNS resolution to ensure that the WSO2 server can reach the endpoint.
3. SSL/TLS Certificate Issues: If the endpoint is secured with SSL/TLS, ensure that the server hosting the endpoint has a valid SSL/TLS certificate installed. If the certificate is self-signed or issued by an untrusted authority, you may need to configure WSO2 to trust the certificate explicitly.
4. Authorization and Authentication Errors: If the endpoint requires authentication or authorization, make sure that the credentials and access permissions provided by WSO2 are correct and sufficient to access the endpoint.
5. Timeout Issues: If the endpoint takes longer to respond, it can lead to timeout errors. Verify the timeout settings in WSO2 and consider increasing them if necessary. Additionally, check the performance and responsiveness of the endpoint itself.
6. Payload or Content-Type Mismatch: If the request payload or the Content-Type header sent by WSO2 does not match the expectations of the endpoint, it can result in errors. Ensure that the payload format and Content-Type header are correctly specified and align with the endpoint's requirements.
7. Backend Service Errors: Endpoint errors can also occur if there are issues with the backend service itself. Verify the logs and error messages from the endpoint to identify any specific errors or exceptions that might provide insights into the problem.

# Fault Sequence

1. Whenever the Back Service Server is down then we get the Endpoint will be Suspended.
2. When there is no Response from the back end up to the Suspension time out.
3. When the Corresponding Drivers not found.
4. Invalid URL.
5. Mediation Errors: If there is an error or exception during any mediation step in the in-flow, such as message transformation, routing, or authentication, the API flow may be directed to the fault sequence. This could be due to misconfigured mediation logic, incorrect message formats, or issues with backend services.
6. This common scenarios like connection timeouts, service unavailability, or backend service exceptions.

# Defining the Fault Sequence

Any Code, if we want to execute it multiple times then we need to write it in a sequence so that the code can be reused and can be efficiently helpful while developing the API or Proxy Services.

### CREATING SEQUENCE :-

Right click on the configs and select new >> sequence.

Name the Sequence and press enter.

Drag and drop a Property Mediator Name the property and give the value as an expression as ‘json-eval($)’ to store all the response from the backend.

Drag and Drop a Log Mediator and PUT all the ERROR\_CODE, ERROR\_MESSAGE , … etc. to trace all the possible error in the log files.

If we want Store the ERRORS In Data Base Drag and drop a DB Report Mediator and write the SQL Statements to Insert the Values into the Data Base.

Next Place a Pay Load Mediator to Display the Error\_Messages to the Client.

Finally Place a respond mediator to send the response to client.

### Sample Sequence Definition :-

A picture containing text, font, number, screenshot

Description automatically generated

### Sequence Configuration.

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

<sequence name=*"Fault\_Sequence"* trace=*"disable"* xmlns=*"http://ws.apache.org/ns/synapse"*>

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

<log description=*"Log error details in case of failure"* level=*"full"*>

<property name=*"MESSAGE"* value=*"An unexpected error occurred."*/>

<property expression=*"$ctx:SYNAPSE\_REST\_API"* name=*"REST\_API"*/>

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

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

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

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

<property expression=*"json-eval($)"* name=*"RESPONSE\_ERROR"*/>

</log>

<dbreport description=*"Insert the ERRORS into DB"*>

<connection>

<pool>

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

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

<user>root</user>

<password>password</password>

</pool>

</connection>

<statement>

<sql><CDATA[insert into API\_STATUS (REST\_API\_NAME, RESPONSE\_DETAILS, ERROR\_CODE, ERROR\_MESSAGE, ERROR\_DETAIL, ERROR\_EXCEPTION) values (?,?,?,?,?,?)]]></sql>

<parameter expression=*"$ctx:SYNAPSE\_REST\_API"* type=*"VARCHAR"*/>

<parameter expression=*"$ctx:RESPONSE\_DETAILS"* type=*"VARCHAR"*/>

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

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

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

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

</statement>

</dbreport>

<payloadFactory description=*"Build payload with error information"* media-type=*"json"*>

<format>{

"MESSAGE": "An unexpected error occurred.",

"REST\_API": "$1",

"ERROR\_CODE": $2,

"ERROR\_MESSAGE": $3,

"ERROR\_DETAIL": $4,

"ERROR\_EXCEPTION": $5

}</format>

<args>

<arg evaluator=*"xml"* expression=*"$ctx:SYNAPSE\_REST\_API"*/>

<arg evaluator=*"xml"* expression=*"$ctx:ERROR\_CODE"*/>

<arg evaluator=*"xml"* expression=*"$ctx:ERROR\_MESSAGE"*/>

<arg evaluator=*"xml"* expression=*"$ctx:ERROR\_DETAIL"*/>

<arg evaluator=*"xml"* expression=*"$ctx:ERROR\_EXCEPTION"*/>

</args>

</payloadFactory>

<respond description=*"Respond back to client with custom error payload"*/>

</sequence>

## Creating the REST API

Right Click on the EsbConfigs Folder and select a new Rest API.

Inside the In sequence Place a Log mediator to know whether the API is hit or not.

Add the Call mediator .

Place an Endpoint whether it is defined or undefined give the necessary configurations like URL and Method like GET, POST, ...etc.

Add a respond mediator at the end.

Add the defined sequence in the fault Sequence.

![A screenshot of a computer

Description automatically generated with low confidence

## Sample API Source Code

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

<api context=*"/errorScenarioAPI"* name=*"Error\_Scenario\_API"* xmlns=*"http://ws.apache.org/ns/synapse"*>

<resource methods=*"GET"*>

<inSequence>

<log level=*"custom"*>

<property expression=*"$ctx:SYNAPSE\_REST\_API"* name=*"API\_NAME"*/>

</log>

<call>

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

</call>

<respond/>

</inSequence>

<outSequence/>

<faultSequence>

<sequence key=*"Fault\_Sequence"*/>

</faultSequence>

</resource>

</api>