



TGH

Making Integrations Simpler



Externalizing Input and Assert Expressions, Handling Error Scenarios, and Parameterization in MUnit

Author
Yuvraj Sinha



Contents

1. Handling Error Scenarios and externalizing input and assert expressions..... 1
2. Parameterization..... 20

Handling error scenarios and externalizing input and assert expressions in MUnit

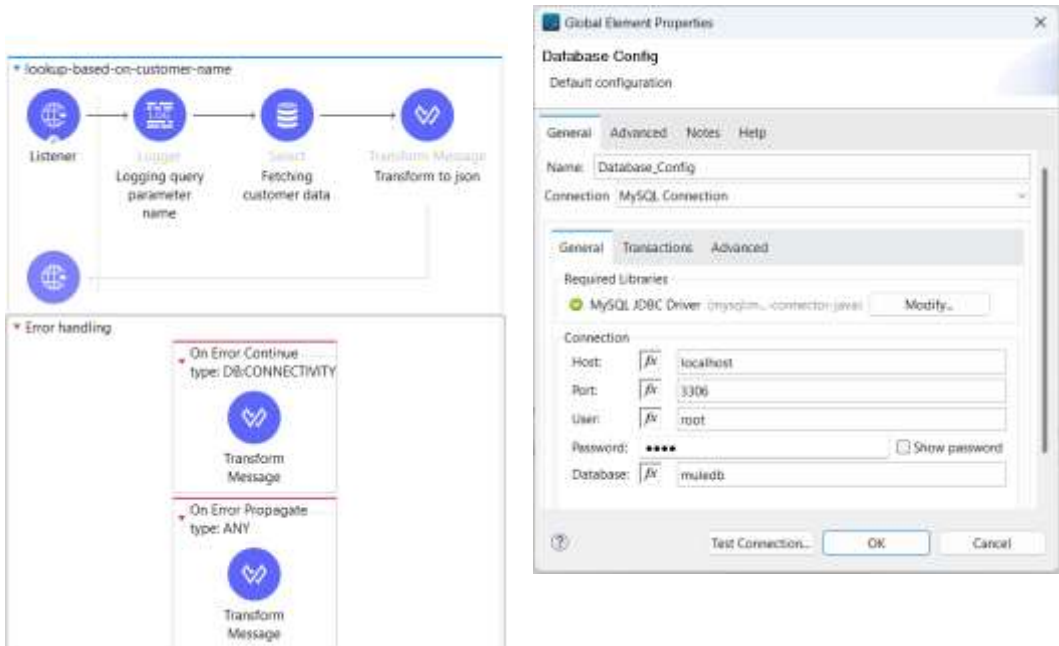
Objective:

We are going to write test cases for the scenarios where the flow is going to throw errors.

The application we are going to test is

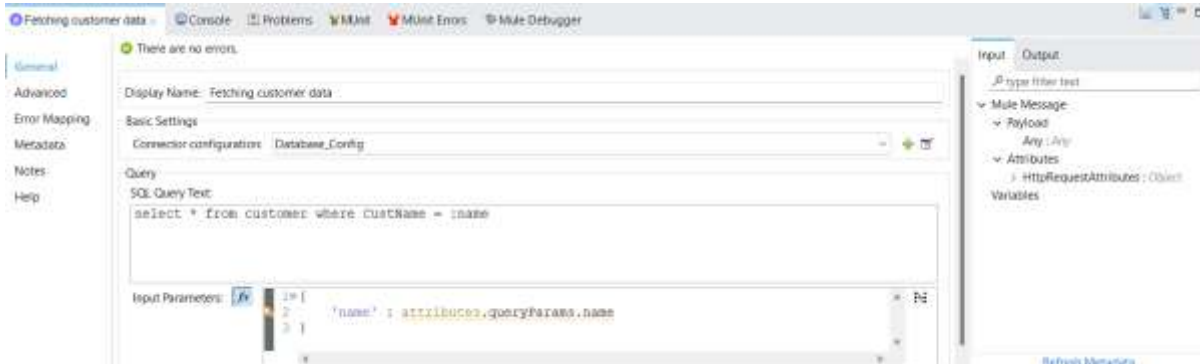
test-case-2

Database configuration



The image shows two screenshots from the MUnit IDE. The left screenshot displays a flowchart for a test case named 'lookup-based-on-customer-name'. The flow starts with a 'Listener' component, followed by a 'Logger' component with the configuration 'Logging query parameter name'. This is followed by a 'Select' component with the configuration 'Fetching customer data', and finally a 'Transform Message' component with the configuration 'Transform to json'. Below the flowchart, the 'Error handling' section is visible, showing two 'On Error' configurations: 'On Error Continue' with type 'DB:CONNECTIVITY' and 'On Error Propagate' with type 'ANY', both leading to 'Transform Message' components. The right screenshot shows the 'Global Element Properties' dialog for a 'Database Config' element. The 'Name' is 'Database_Config' and the 'Connection' is 'MySQL Connection'. The 'General' tab is active, showing 'Required Libraries' as 'MySQL JDBC Driver (mysql-connector-java)' and 'Connection' details: Host: localhost, Port: 3306, User: root, Password: masked, and Database: muledb. There are 'Test Connection...', 'OK', and 'Cancel' buttons at the bottom.

Query



The image shows a screenshot of the MUnit IDE interface. The 'Fetching customer data' component is selected, and the 'Query' tab is active. The 'SQL Query Text' field contains the query: `select * from customer where custName = :name`. The 'Input Parameters' section shows a table with the following data:

Index	Value
1	name
2	attributes.queryParams.name
3	

The 'Console' tab shows 'There are no errors.' The 'Input/Output' tab on the right shows the 'Payload' as 'Any :Any' and 'Attributes' as 'HttpRequestAttributes : Object' and 'Variables'.

Database

9 • `SELECT * FROM muledb.customer;`

Result Grid | Filter Rows:

	CustId	CustName	OrderQty
▶	100	Yuvraj	10
	101	Adrish	20
	102	Yuvraj	5
•	NULL	NULL	NULL

Postman Request and response

GET `http://localhost:8081/test-case-2?name=Yuvraj` Send

Params • Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Description
name	Yuvraj	

Body Cookies Headers (3) Test Results Status: 200 OK, Time: 34 ms, Size: 275 B Show as example

Pretty Raw Preview Visualize JSON Copy Search

```

1 [
2   {
3     "CustId": 100,
4     "OrderQty": "10",
5     "CustName": "Yuvraj"
6   },
7   {
8     "CustId": 102,
9     "OrderQty": "5",
10    "CustName": "Yuvraj"
11  }
12 ]
  
```

Error handler “transform messages”

DB:CONNECTIVITY

```

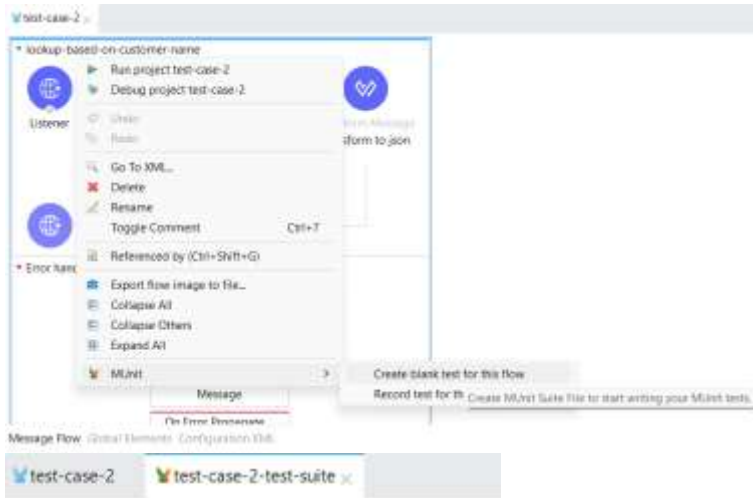
Output Payload
1= %dw 2.0
2 output application/json
3 ---
4 {
5   Status : 200,
6   Message : "cannot connect to database"
7 }
8
  
```

MULE:ANY

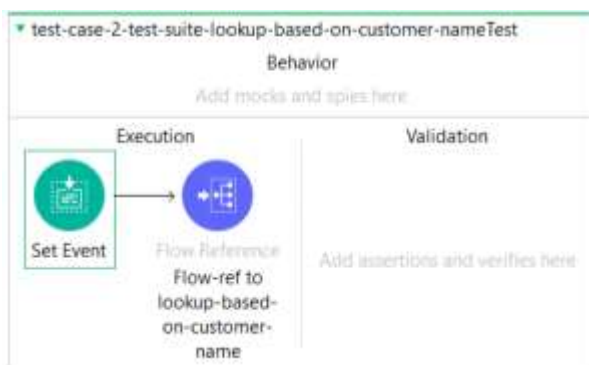
```

Output Payload
1= %dw 2.0
2 output application/json
3 ---
4 {
5   Status : 500,
6   Message : "MULE:ANY"
7 }
  
```

Step 1: Select the flow you want to test (“lookup-based-on-customer-name”) then right-click on the flow and click on “MUnit” then “Create blank test for this flow”. This creates a blank test flow where you can write your test cases and also it creates a test suite.



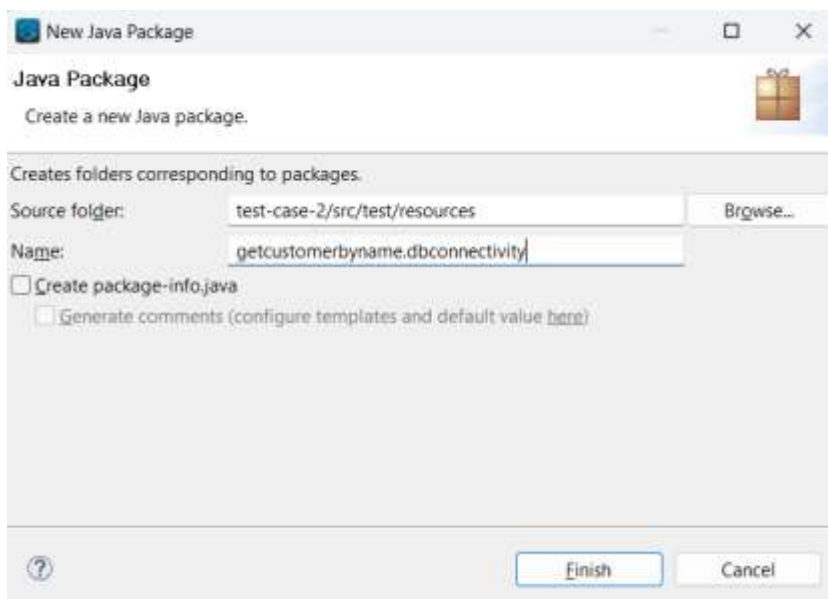
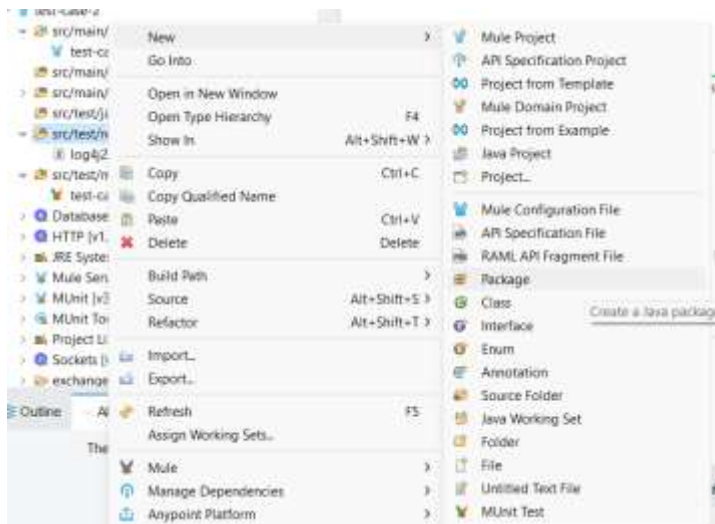
Step 2: Now drag and drop a “Set Event” processor from the “MUnit” module in the mule pallet before the flow reference in the execution part of the test flow.



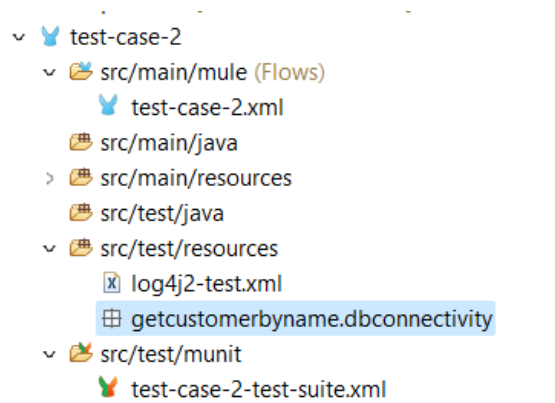
Step 3: Now we need to externalize the values we are going to set in the “Set event”. We need to create a package in “src/test/resources”.

Right-click on “src/test/resources” and then on “new” and then on “package”.

©TGH Software Solutions Pvt. Ltd.

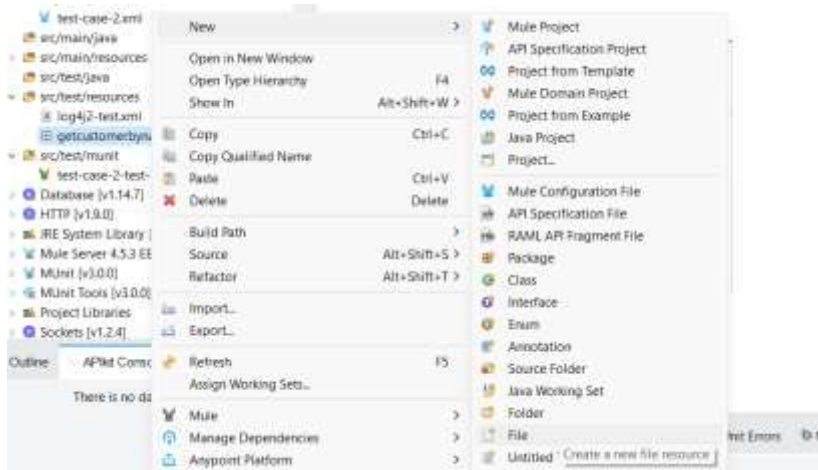


Put the package name as “**getcustomerbyname.dbconnectivity**” and click on finish.

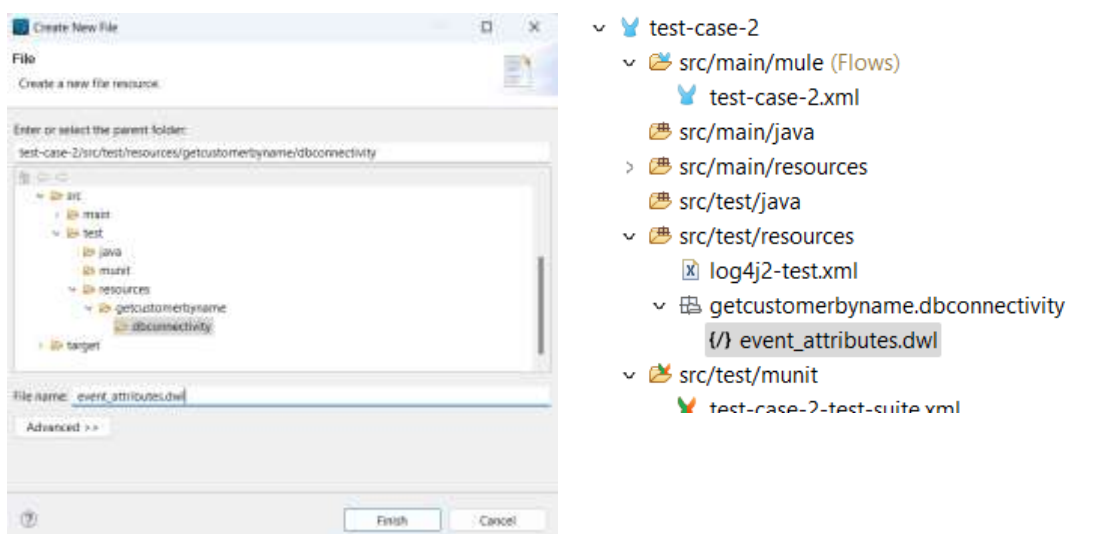


Step 4: Now right-click on the package “getcustomerbyname.dbconnectivity” then “New” and then file

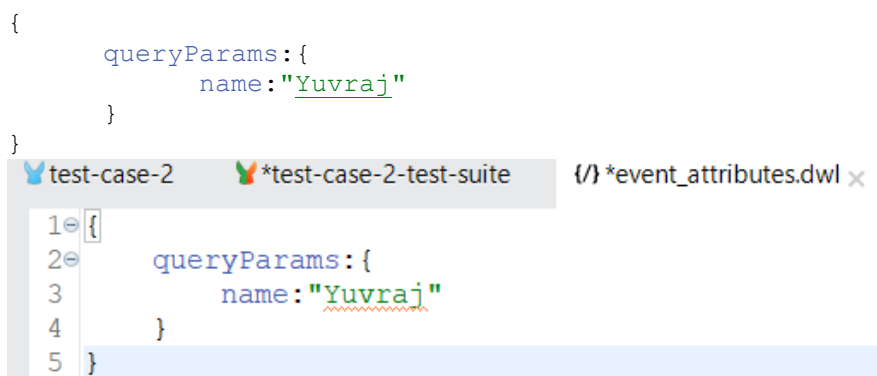
©TGH Software Solutions Pvt. Ltd.



Then give the file name as “event_attributes.dwl” and click finish.



Now double-click on the “event_attributes.dwl” it opens the file. Paste this expression below



And save the file.

Step 5: Now we need to refer to this in the “Set event” processors attributes section.

Now go to the attributes section in the “Set event” processor and click on the expression mode in “value” (highlighted as 1) then click on the button highlighted as 2.

©TGH Software Solutions Pvt. Ltd.



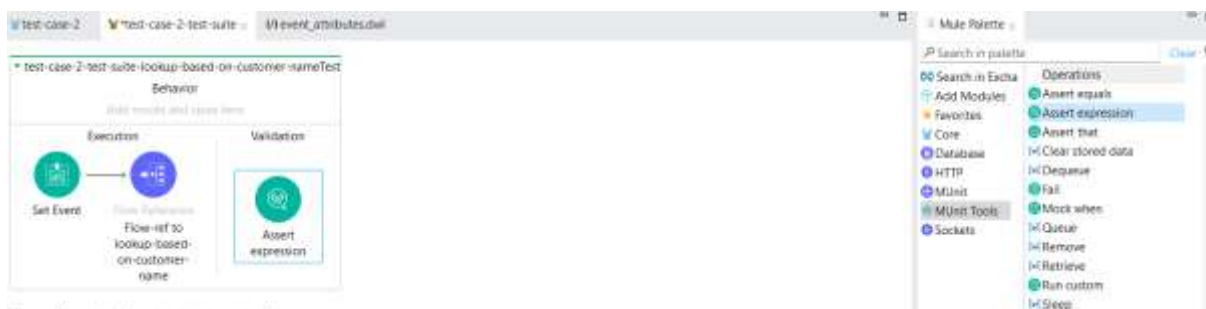
Then copy and paste this expression

```
readUrl("classpath://getcustomerbyname/dbconnectivity/event_attributes.dwl")
```



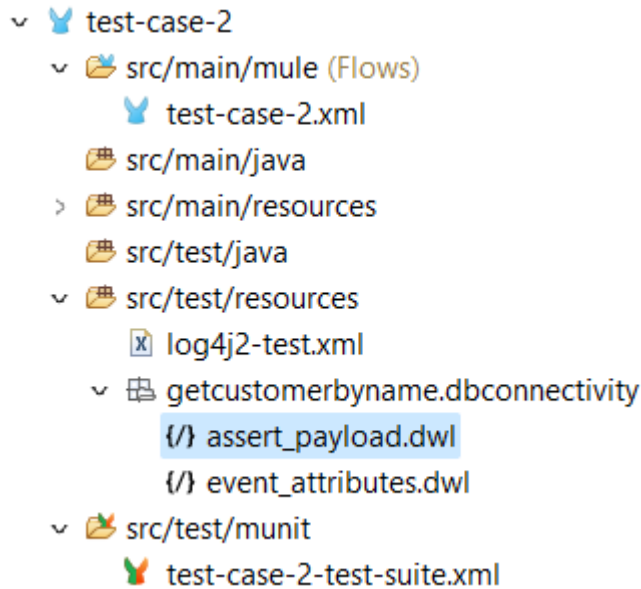
Then click on done

Step 5: Now drag and drop an “Assert Expression” from the “MUnit Tools” module in the mule pallet to the “Validation” section of the test flow.



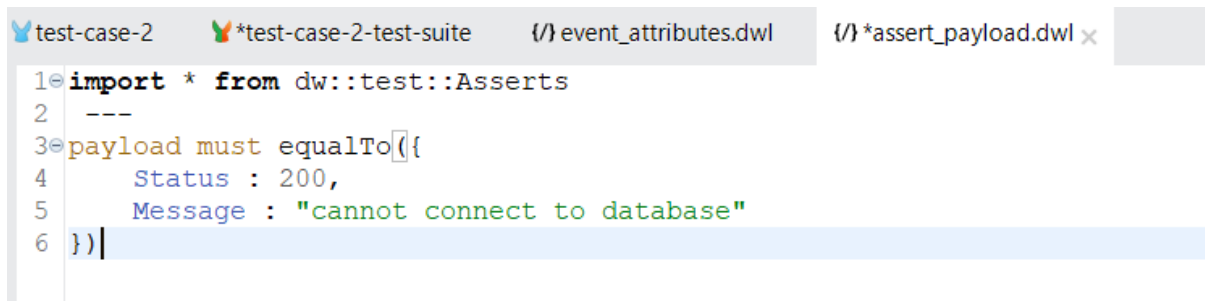
Step 6: Now we need to externalize the expression we are going to assert

Now create a file with the name “assert_payload.dwl” in “getcustomerbyname.dbconnectivity”



Open the “assert_payload.dwl” file and paste the expression below

```
import * from dw::test::Asserts
---
payload must equalTo({
  Status: 200,
  Message: "cannot connect to database"
})
```



Now save the file

Step 7: Now we need to refer to this file in “Assert Expression”

Copy and paste the expression below expression in “Expression” section of assert expression.

```
import getcustomerbyname::dbconnectivity::assert_payload
---
assert_payload::main({payload:payload, attributes:attributes, vars:vars})
```

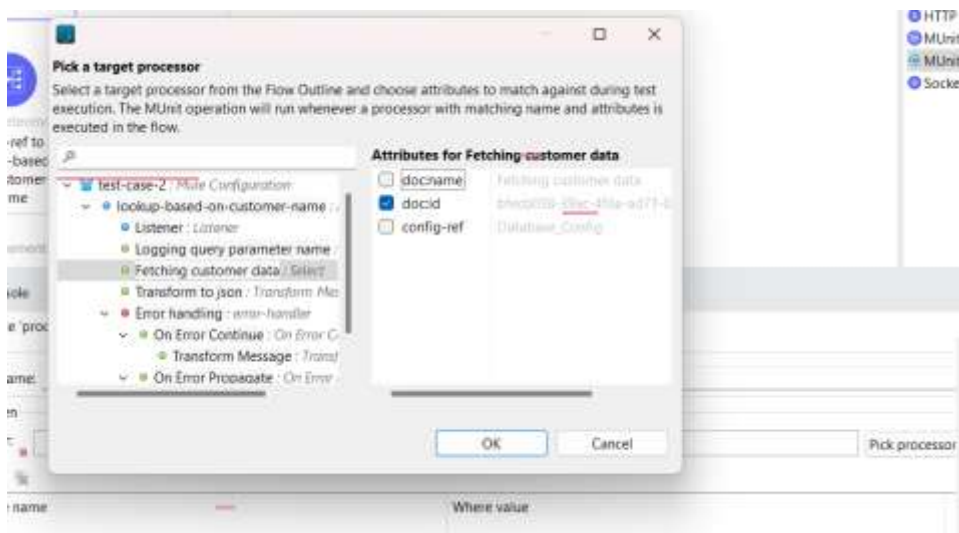


Step 8: Now we need to mock the database “component” to throw a “DB:CONNECTIVITY” error.

Drag and drop a “Mock when” in the “Behavior” section of the test flow from the “MUnit Tools” module in the mule pallet.



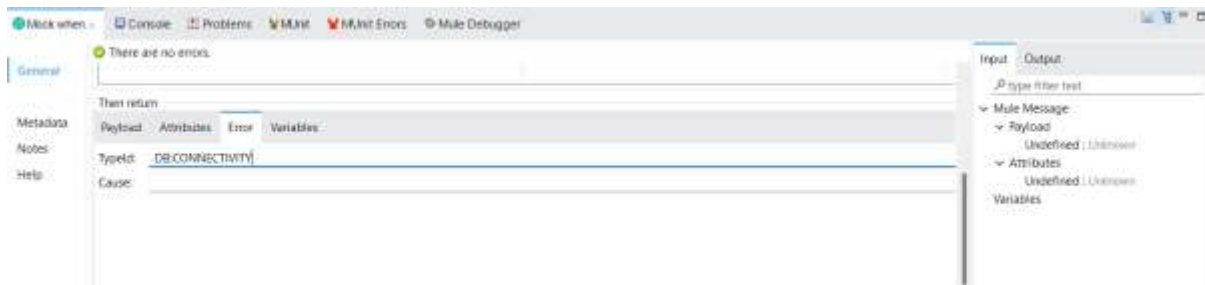
Step 9: Now go to mock when and click on “Pick processor” then select “fetching customer data” and then select “doc:id”



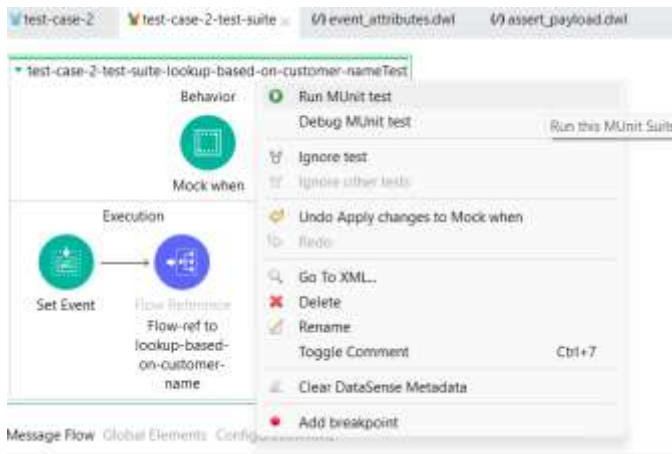
Then click on “OK”

Step 10: Then go to the “Error” section in “Then return” in “Mock when” and paste the expression below in “TypeId”

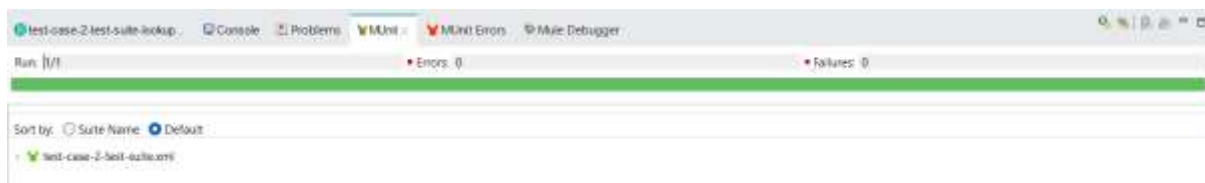
DB:CONNECTIVITY



Step 10: Now save the project and we need to run the test case. Double-click on the test flow right-click and then click on “Run MUnit test”.



You will see the test case will successfully pass.

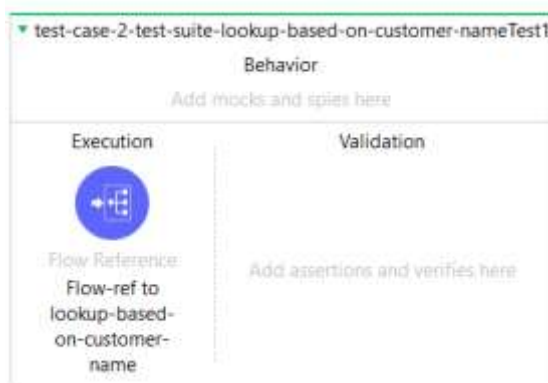
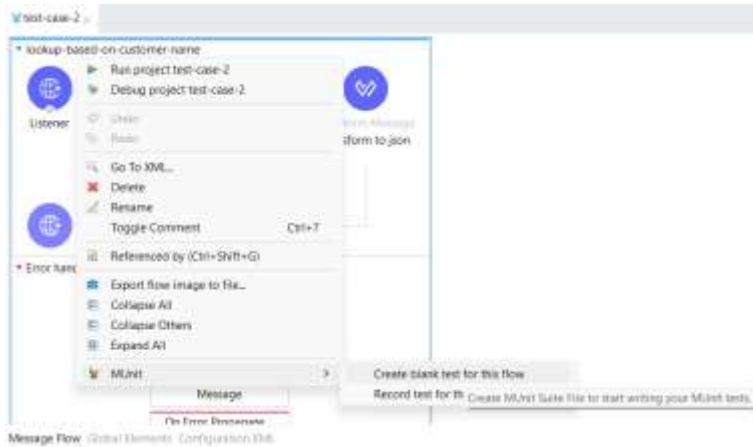


In the above scenario, we are testing the error which is handled by “on error continue”. Now in the second scenario, we are going to test the errors that are handled by “on error propagate”.

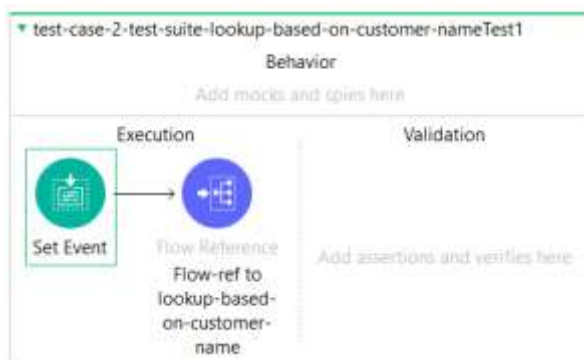
Objective:

Writing the test cases for the scenario where the error is handled by “on error propagate”.

Step 1: Select the flow you want to test (“lookup-based-on-customer-name”) then right-click on the flow and click on “MUnit” then “Create blank test for this flow”.



Step 2: Now drag and drop a “Set Event” processor from the “MUnit” module in the mule pallet before the flow reference in the execution part of the test flow.



Step 3: Now we need to externalize the values we are going to set in the “Set event”. Create a package with the name “getcustomerbyname.any” in “src/test/resources”

(If you forgot the steps refer to the steps of the above scenario)

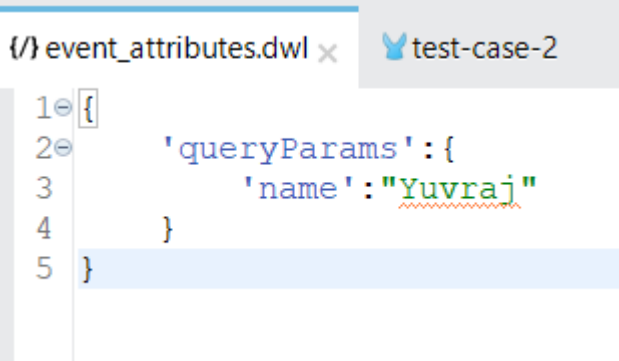
- test-case-2
 - src/main/mule (Flows)
 - test-case-2.xml
 - src/main/java
 - src/main/resources
 - src/test/java
 - src/test/resources
 - log4j2-test.xml
 - getcustomerbyname.any
 - getcustomerbyname.dbconnectivity
 - assert_payload.dwl
 - event_attributes.dwl

Step 4: Now create a file with name “event_attributes.dwl” in the package “getcustomerbyname.any” .

- test-case-2
 - src/main/mule (Flows)
 - test-case-2.xml
 - src/main/java
 - src/main/resources
 - src/test/java
 - src/test/resources
 - log4j2-test.xml
 - getcustomerbyname.any
 - event_attributes.dwl
 - getcustomerbyname.dbconnectivity

Now paste the expression below into that file and save it.

```
{  
  'queryParams': {  
    'name': "Yuvraj"  
  }  
}
```

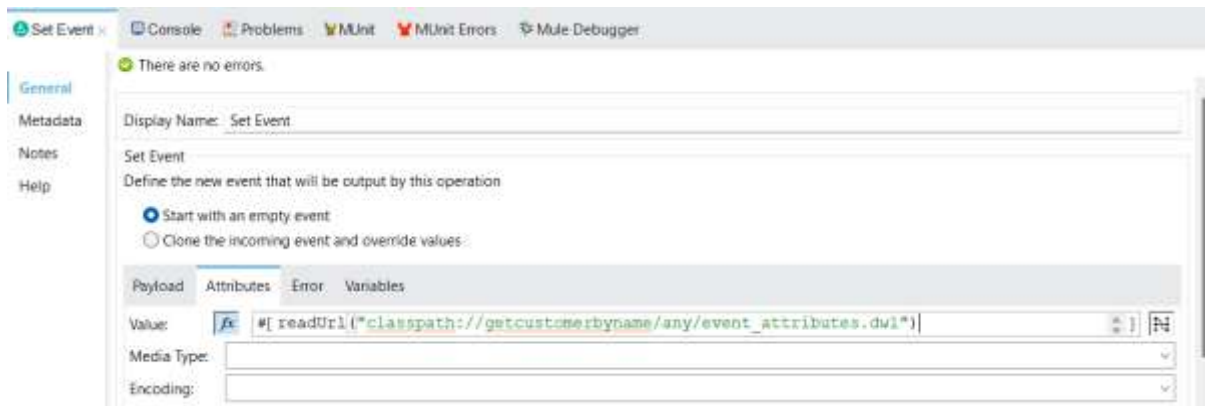


```
{ } event_attributes.dwl x test-case-2  
1 {  
2   'queryParams': {  
3     'name': "Yuvraj"  
4   }  
5 }
```

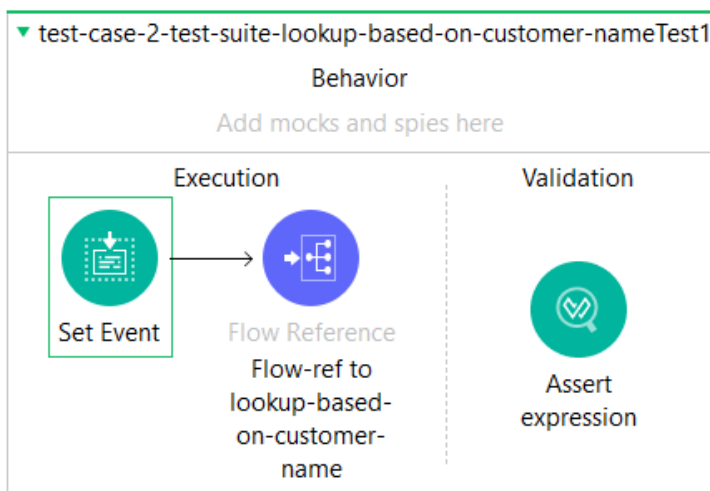
Step 5: Now we need to refer to this file in the attributes section of the “Set Event” processor

Paste the below expression in the attributes section of “Set event” processor.

```
readUrl ("classpath://getcustomerbyname/any/event_attributes.dwl")
```

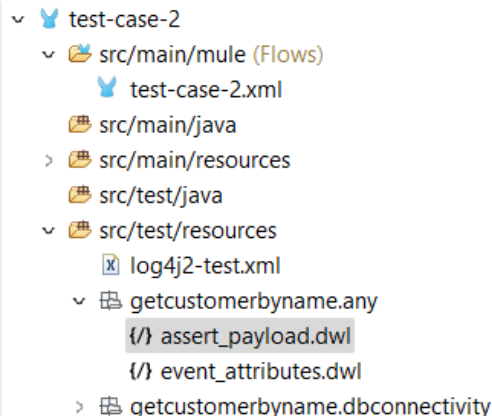


Step 6: Now drag and drop an “Assert Expression” in the “validation” section of the test flow



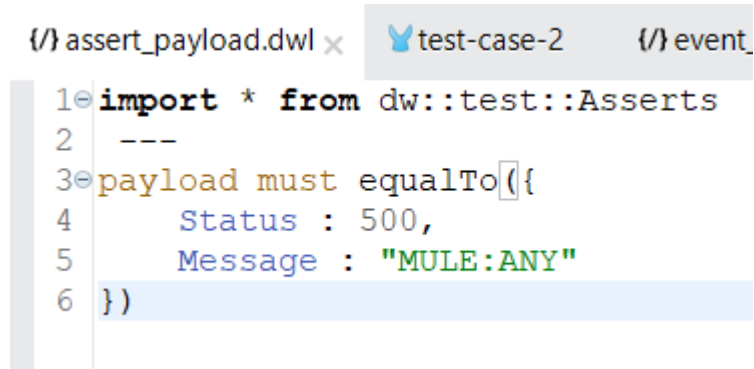
Step 7: Now we need to externalize the expression we are going to assert

Now create a file with the name “assert_payload.dwl” in “getcustomerbyname.any”



Step 8: Now paste the below expression into that file and save it.

```
import * from dw::test::Asserts
---
payload must equalTo({
  Status : 500,
  Message : "MULE:ANY"
})
```



Step 9: Now we need to refer to this file in “Assert Expression”

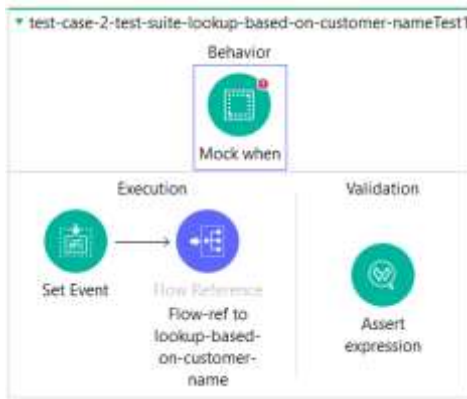
Copy and paste the expression below expression in the “Expression” section of assert expression.

```
import getcustomerbyname::any::assert_payload
---
assert_payload::main({payload:payload,attributes:attributes,vars:vars})
```

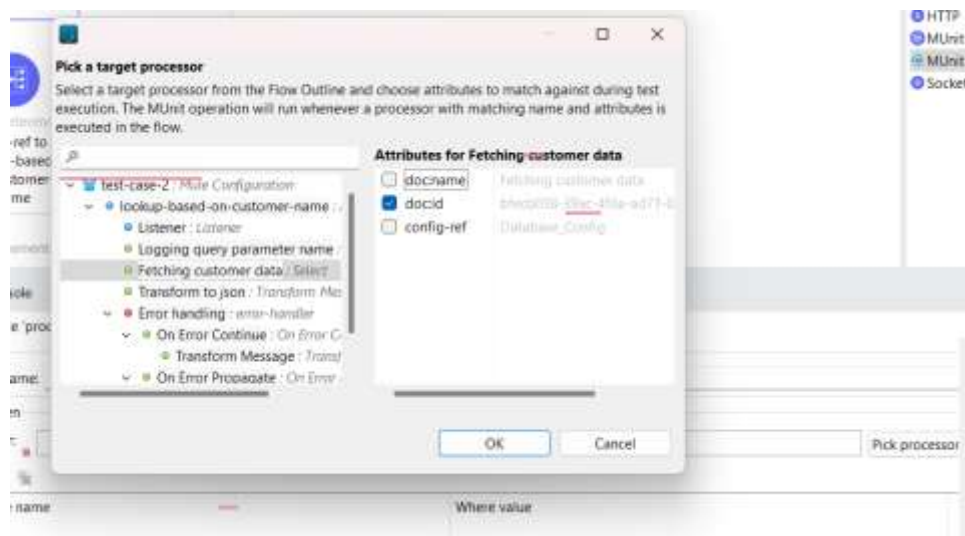


Step 10: Now we need to mock the database “component” to throw a “MULE:ANY” error.

Drag and drop a “Mock when” in the “Behavior” section of the test flow from the “MUnit Tools” module in the mule pallet.



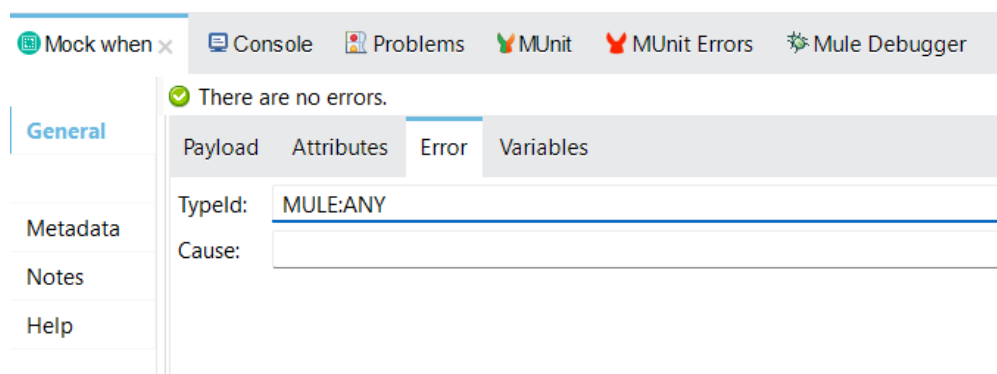
Step 11: Now go to “Mock when” and click on “Pick processor” then select “fetching customer data” and then select “doc:id”



Then click on “OK”

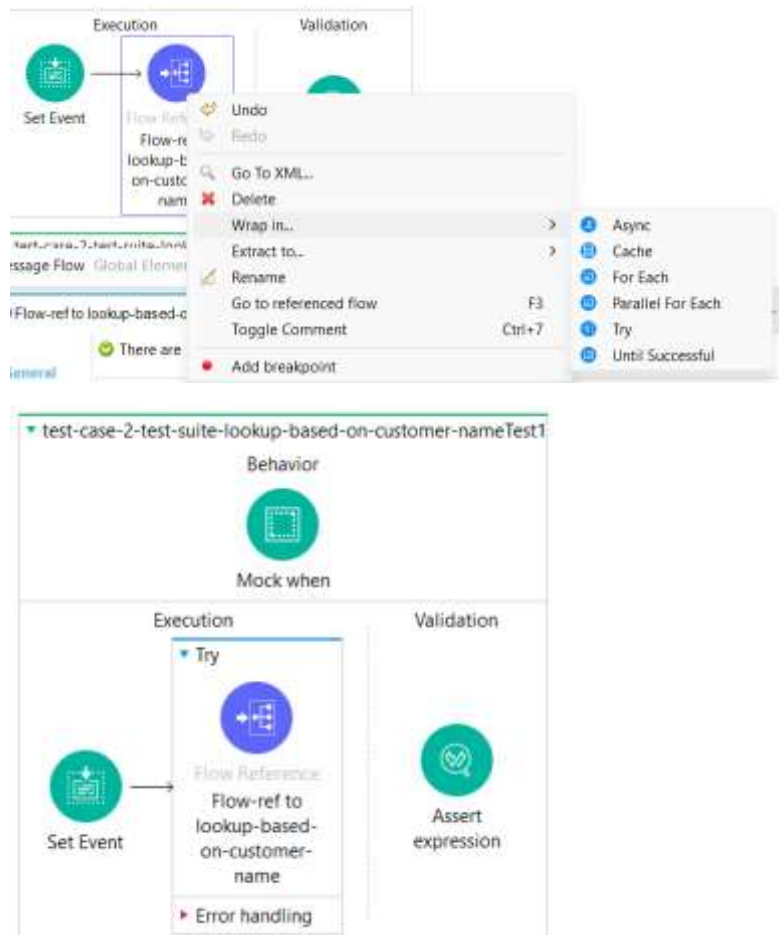
Step 12: Then go to the “Error” section in “Then return” in “Mock when” and paste the expression below in “TypeId”

MULE:ANY

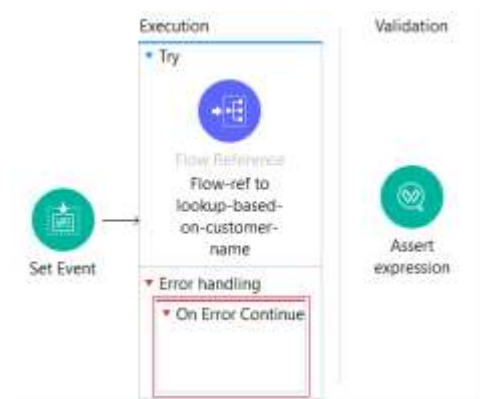


Step 13: Now we need to wrap the flow reference in try-catch.

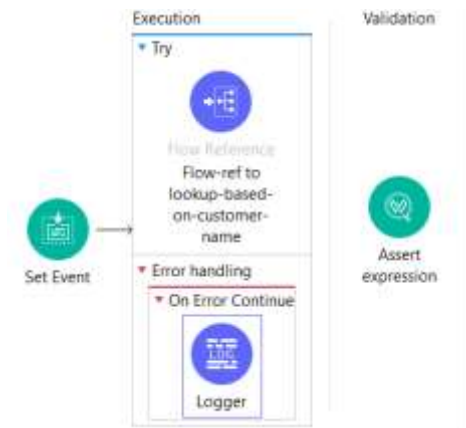
Right-click on the flow reference and go to wrap in then click on try.



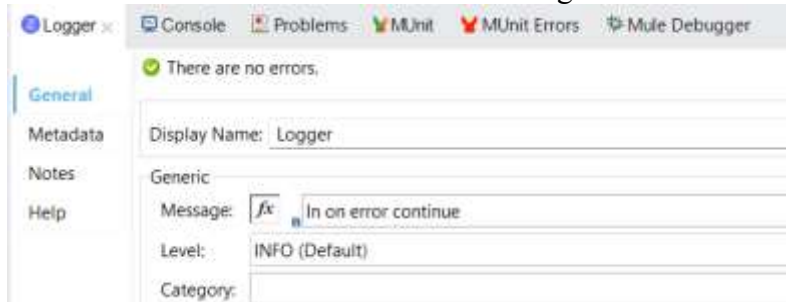
Step 14: Now drag and drop a on error continue on the error handling section of the try-catch



Step 15: Now drag and drop a logger on the “On Error Continue” section.

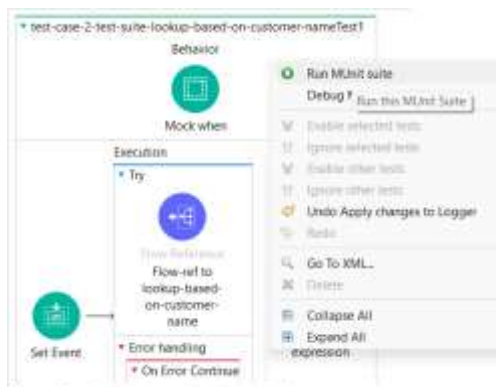


Now write “In on error continue” in the message section of the logger

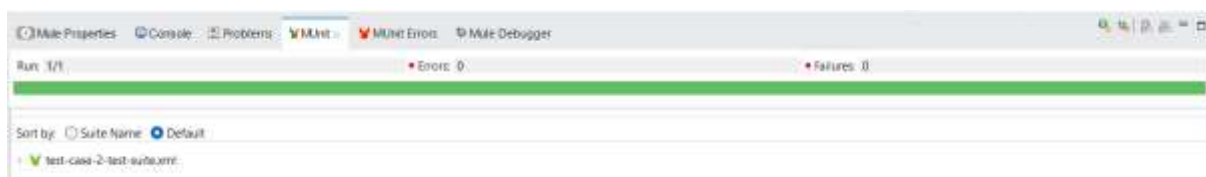


(We are wrapping up the flow reference in try-catch because the error thrown in the main flow will be propagated to the flow reference because it is handled in “on error propagate” so the control will not go to “Assert expression” if not wrapped in try-catch)

Step 16: Now save the project and we need to run the test case. Double-click on the test flow right-click and then click on “Run MUnit test”.



You will see the test case will successfully pass.

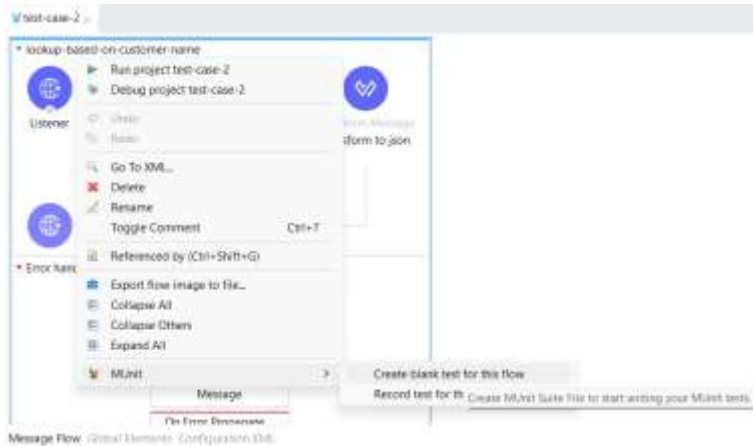


Parameterization

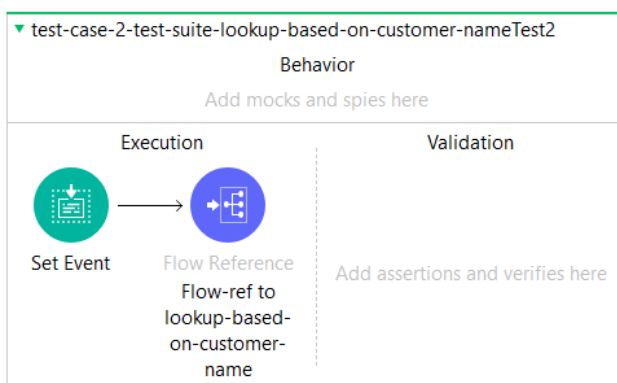
Objective:

We are going to parameterize the event attributes and the expression we want to assert.

Step 1: Select the flow you want to test (“lookup-based-on-customer-name”) then right-click on the flow and click on “MUnit” then “Create blank test for this flow”.



Step 2: Now drag and drop a “Set Event” processor from the “MUnit” module in the mule pallet before the flow reference in the execution part of the test flow.



Step 3: To parameterize the attributes we need to write the expression below

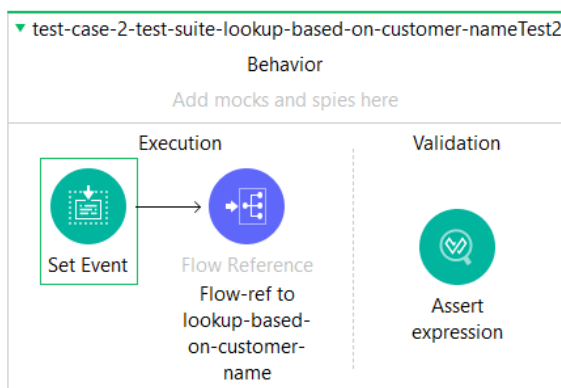
```
{'queryParams':{'name': Mule::p('custname')}}
```

(p) function returns a string that identifies the value of one of these input properties: Mule property placeholders, System properties, or Environment variables)

Copy and paste the above expression in the “Attributes” section of the “Set Event”



Step 4: Now drag and drop an “Assert Expression” in the “validation” section of the test flow

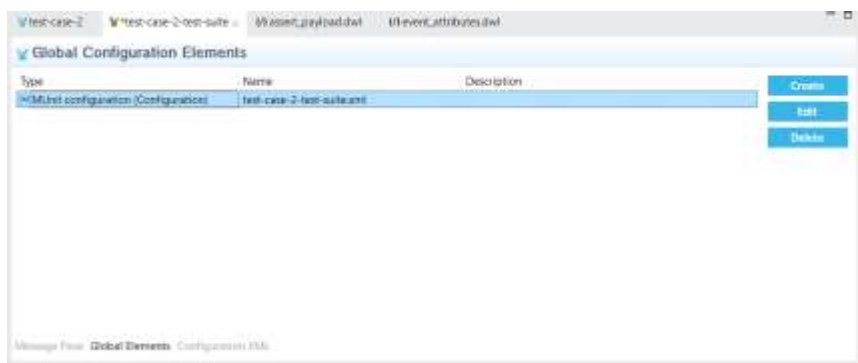


Step 5: Now paste the below expression in the “Expression” section in “Assert expression”

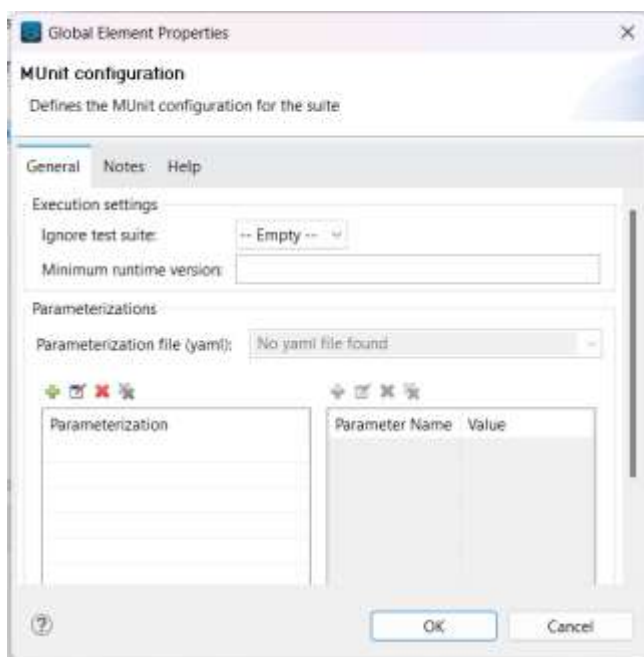
```
import * from dw::test::Asserts
---
payload must haveSize(Mule::p('size')) as Number)
```



Step 6: Now we need to pass the parameters we mentioned in the p() function. Now go to “Global Elements” and select “MUnit Configuration” and click on Edit

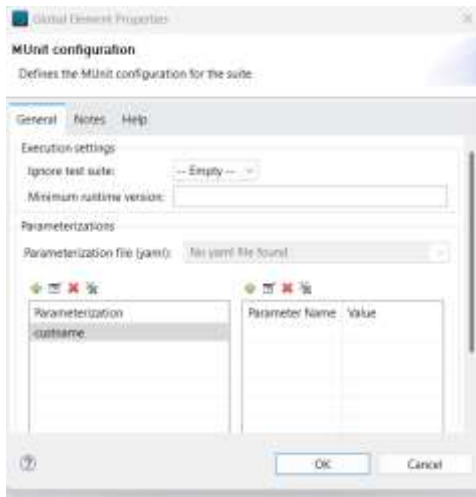
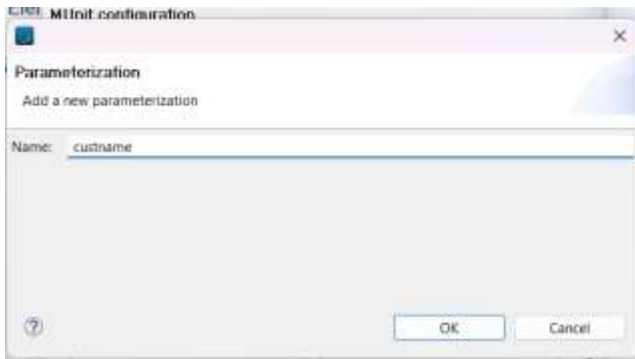


This window will pop up

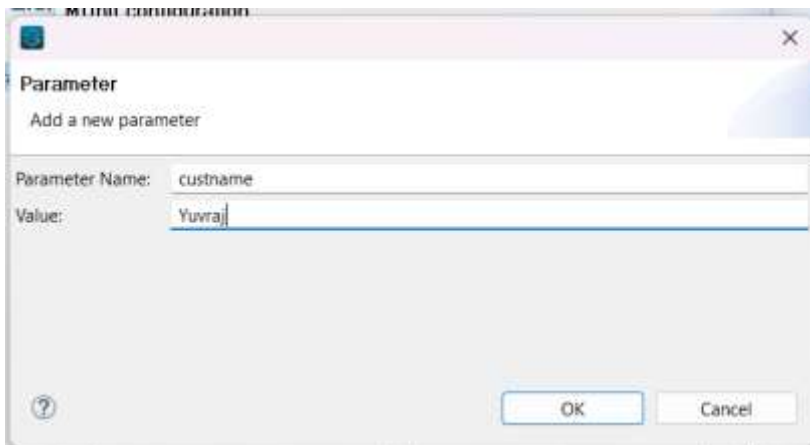


Step 7: click on the “+” button in the “Parameterization” section give the parameterization name as custname and click on “OK” (It can be any name under this we are going to define groups of parameters which we will pass)

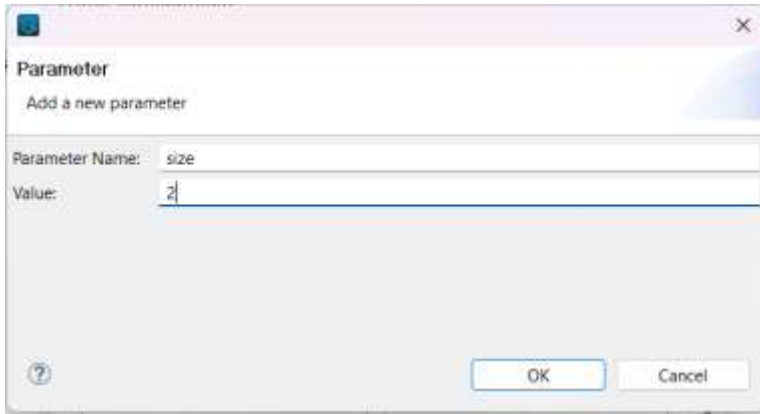
©TGH Software Solutions Pvt. Ltd.



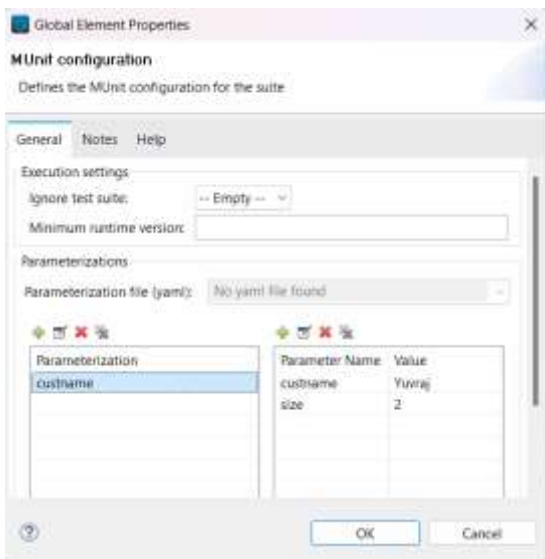
Step 8: Now select custname and click on the “+” button on the “Parameter name and value” section. Add “Parameter name” as custname and “Value” as Yuvraj and click on “OK”.



Step 9: Now add another parameter with “Parameter name” as size and “Value” as 2 and click on “OK”.



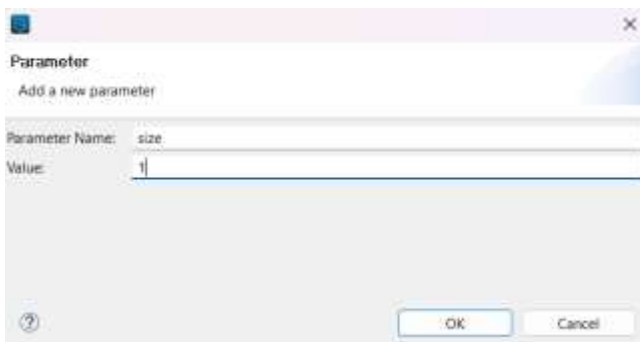
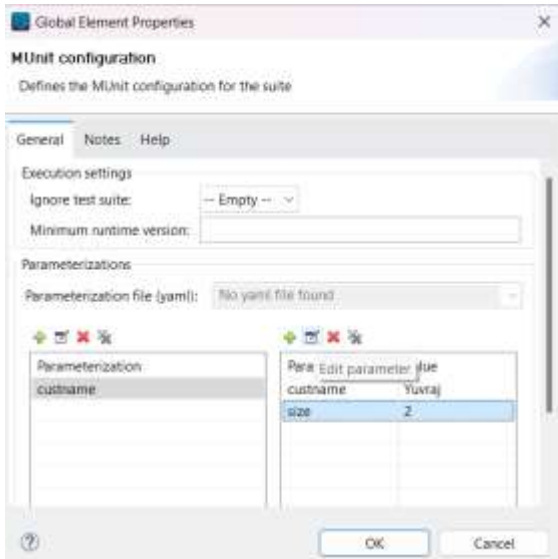
Step 10: Click “OK” In the Global Element Properties window



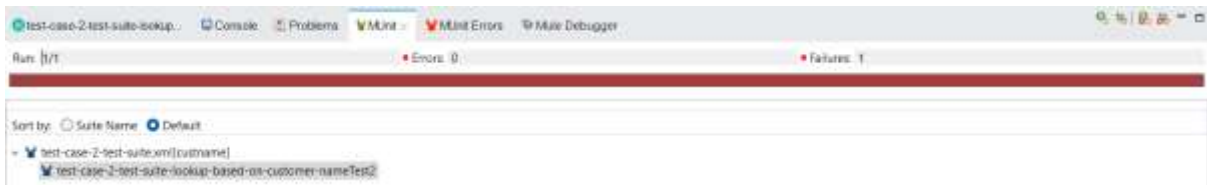
Step 10: Now you are going to see some errors in the “MUnit Configuration”. This is occurring due to a bug in Anypoint Studio. Go to configuration XML you can see `<munit:parameterization>` is repeating.



The tags highlighted in red are repeating. Now remove the tags and the errors will be gone. (The number of elements repeating may differ, just removing the duplicates will solve the issue)

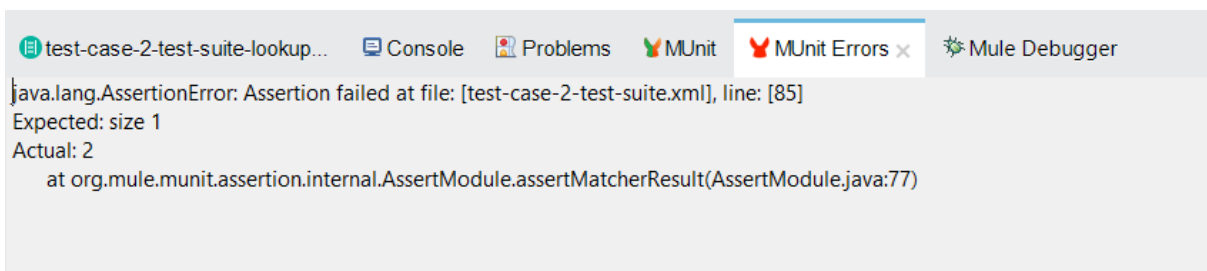


Now click on “OK” and again click “OK” in “Global Element Properties” save the project and run the test case. You will see the test case fails.



This is because with the query parameter “Yuvraj” there are two 2 records in the database and we are giving the size as “1”. Previously we gave the size as 2 that’s why it passed.

You can see the reason in the error section





TGH

Making Integrations Simpler

TGH Software Solutions Pvt. Ltd.

www.techygeekhub.com

At TGH, we specialize in driving digital transformation through seamless Integration Technologies.

Operating as an INTEGRATION FACTORY, we serve as a one-stop shop for all your integration needs. Our expert team is well-versed in enterprise software and legacy system integration, along with leading iPaaS technologies like Boomi, MuleSoft, Workato, OIC, and more.

We're committed to enhancing business processes and solving problems through our integration expertise.



Email address

connect@techygeekhub.com



Phone number

+ 011-40071137
+ 91-8810610395



Our offices

Noida Office

iThum
Plot No -40, Tower A,
Office No: 712,
Sector-62, Noida,
Uttar Pradesh, 201301

Hyderabad Office

Plot no: 6/3, 5th Floor,
Techno Pearl Building,
HUDA Techno Enclave,
HITEC City, Hyderabad,
Telangana 500081

