Kutumba Department Requirement-Decoded and Procedure:-

**Concept Behind the Requirement:**

Department requested "ESB" to find a way where clients can Access the Api's By Avoiding the CodeRedeployement Procedure:::::::::As well as Providing Security for the Api’s hosted on the MuleSoft.

1.Maintaining a DataBase tables to Store the Clients,update/insert the clients, So that the entire control will be taken by the Department.

2.Even Feilds table is also maintained to Expose only particular feilds to particular clients.

Requirement Achievement is Done -

Procedure Followed to Achieve the Scenario:

-> Created Table esb\_Schema.Clients where it holds the

1.Client-Codes

2.StartTime

3.EndTime

4.APICodes(Which Api's need to be provided to which-Client)

-> Created Table esb\_Schema.ApiMaster where it holds the APInames and API-Id's.

-> Created Table esb\_Schema.datafeilds where it holds the DataFeilds of one Api "GetSchemes API".

The Api Flow is Planned in Such a Way that...

#.Client need to be validated:

The ClientCode which is Coming inside the Flow is captured first, and sent to the Validation. The clientcode entering into API-Flow gets checked in the Database first, which means if that client exists in the database then the flow should continue. Else there will be a Error, The Client-Secret provided is Invalid.

#.Time Validation:

There will be StartTime and Endtime in the Table. The now() time gets Compared with the DataweaveScript. If the Validation is true, the flow continues orelse a error will be raised Saying "UN-Permitted-TimeIntervals".

timevalidation-dataweave Script:

%dw 2.0

var startTime = payload.StartTime as Time

var endTime = payload.EndTime as Time

fun validate(time: Time) =

(((time - startTime) as Number > 0) and ((endTime - time) as Number > 0))

output application/json

---

{

validation: validate((now() >> "IST") [11 to 18] as Time),

startTime: startTime ,

endTime: endTime ,

IST: (now() >> "IST") [11 to 18],

istastime: (now() >> "IST") [11 to 18] as Time

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In the Above Dw Code the now time gets splitted as 00:00:00Z and then gets validated,,,The result might be True or False. If “true” the flow continues. If it's false then as mentioned above the error is Raised....

#.Client and API validation:

This Step deals with the Stored-Variable APIID from the esb\_Schema.Clients in which it has APiid's Assosciated with the particlar client and the ApiId from the esb\_Schema.ApiMaster table (for this a select query is fired with the Api-name which returns the APIid into the flow). Now if the Api-id matches with the API'ds then the flow continues orelse error is raised.

%dw 2.0

output application/json

---

{

"validator": (((vars.APIID splitBy ",")[0 to -1]) contains vars.CapturedDataBaseAPIID)as String

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The above logic splits the APIID forms an Array ....And the comparison be like, if the CapturedDataBaseAPIID exists in the Array(vars.APIID) then the output is “True”, the flow continues or else if output is “False” the flow will Raise the Error that "The Client Has No Permission to Invoke the API."

#.Api Invoking Will be done and goes various Data Decryption Procedures and final step will be Feilds exposure.

#.Feilds\_Exposure:

A Table esb\_Schema.datafeilds is created for each APi, to expose which feilds need to be exposed for which client

refer the table for the reference.

Based on the clientId the Assosciated feilds for that particular client is allowed to get into the flow and

the below logic is applied to split and convert the feilds into an Array

%dw 2.0

output application/json

---

{

"Client\_id-DB": payload.ClientId reduce $,

"FeildList": ((payload.OutputFeilds reduce $)splitBy ",")[0 to -1]

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Finally a Feild Exposure Control logic is written like this to expose if it exist for that particular client

%dw 2.0

**output** **application/json** skipNullOn="everywhere"

**var** APiResponse = vars.APiResponse

**fun** check(respData)= **if**((respData matches "Resident\_ID") **and** (payload.FeildList contains "Resident\_ID")) (APiResponse.ResultDataList reduce $).Resident\_ID

**else** **if**((respData matches "Benf\_Name") **and** (payload.FeildList contains "Benf\_Name")) (APiResponse.ResultDataList reduce $).Benf\_Name

**else** **if**((respData matches "District\_Name") **and** (payload.FeildList contains "District\_Name")) (APiResponse.ResultDataList reduce $).District\_Name

**else** **if**((respData matches "Taluk\_Name") **and** (payload.FeildList contains "Taluk\_Name")) (APiResponse.ResultDataList reduce $).Taluk\_Name

**else** **if**((respData matches "Dept\_Name") **and** (payload.FeildList contains "Dept\_Name")) (APiResponse.ResultDataList reduce $).Dept\_Name

**else** **if**((respData matches "Scheme\_Name") **and** (payload.FeildList contains "Scheme\_Name")) (APiResponse.ResultDataList reduce $).Scheme\_Name

**else** **if**((respData matches "Amount\_Received") **and** (payload.FeildList contains "Amount\_Received")) (APiResponse.ResultDataList reduce $).Amount\_Received

**else** **if**((respData matches "Payment\_Received\_Year") **and** (payload.FeildList contains "Payment\_Received\_Year")) (APiResponse.ResultDataList reduce $).Payment\_Received\_Year

**else** **null**

---

{

StatusCode: APiResponse.StatusCode,

StatusText: APiResponse.StatusText,

Response\_ID: APiResponse.Response\_ID,

Request\_ID: APiResponse.Request\_ID,

ResultDataList: [

{

"Resident\_ID": check("Resident\_ID"),

"Benf\_Name": check("Benf\_Name"),

"District\_Name": check("District\_Name"),

"Taluk\_Name": check("Taluk\_Name"),

"Dept\_Name": check("Dept\_Name"),

"Scheme\_Name": check("Scheme\_Name"),

"Amount\_Received": check("Amount\_Received"),

"Payment\_Received\_Year": check("Payment\_Received\_Year")

}

]

}



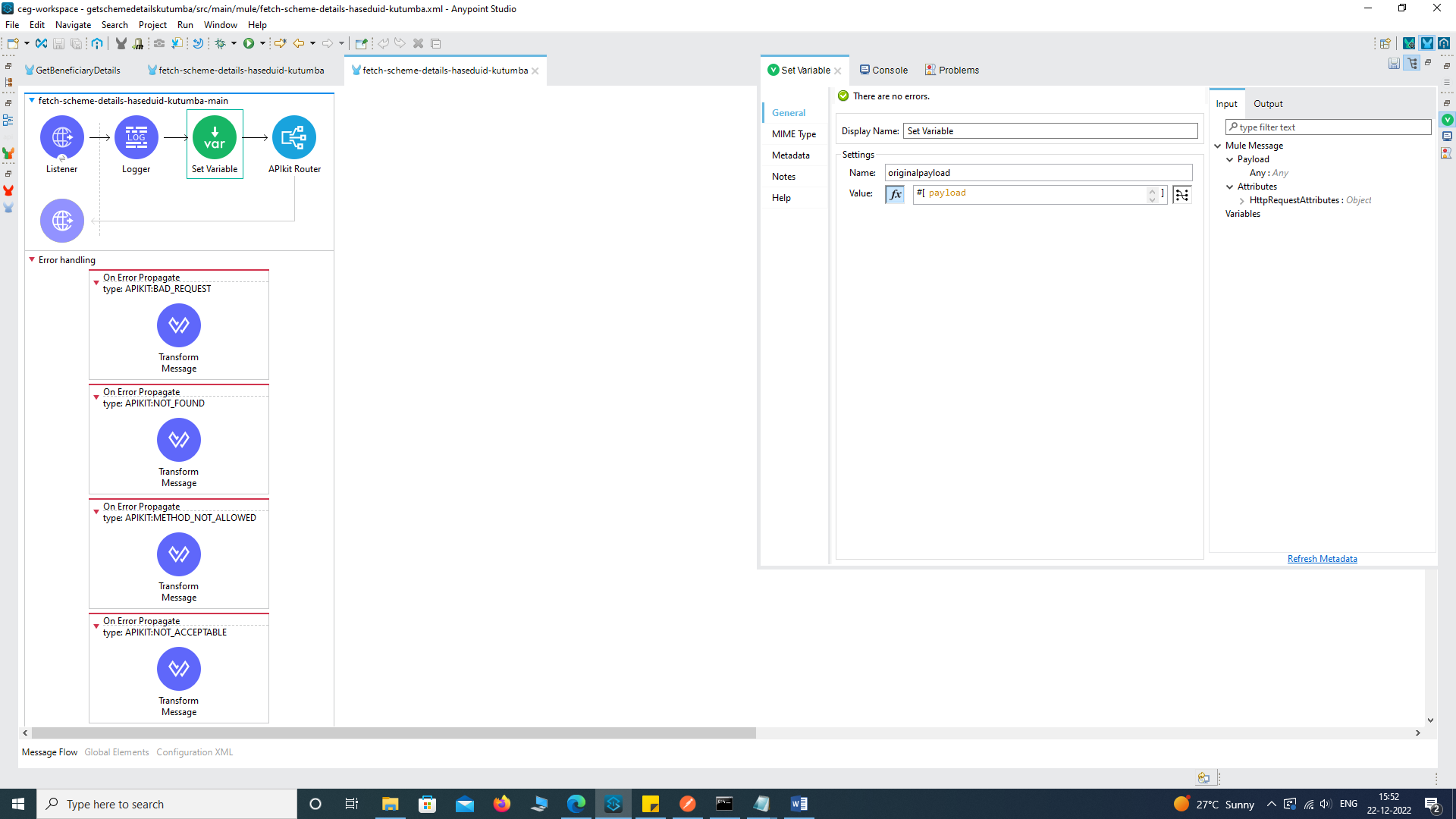
-----------------------------------------------------------------------------------------------------------------

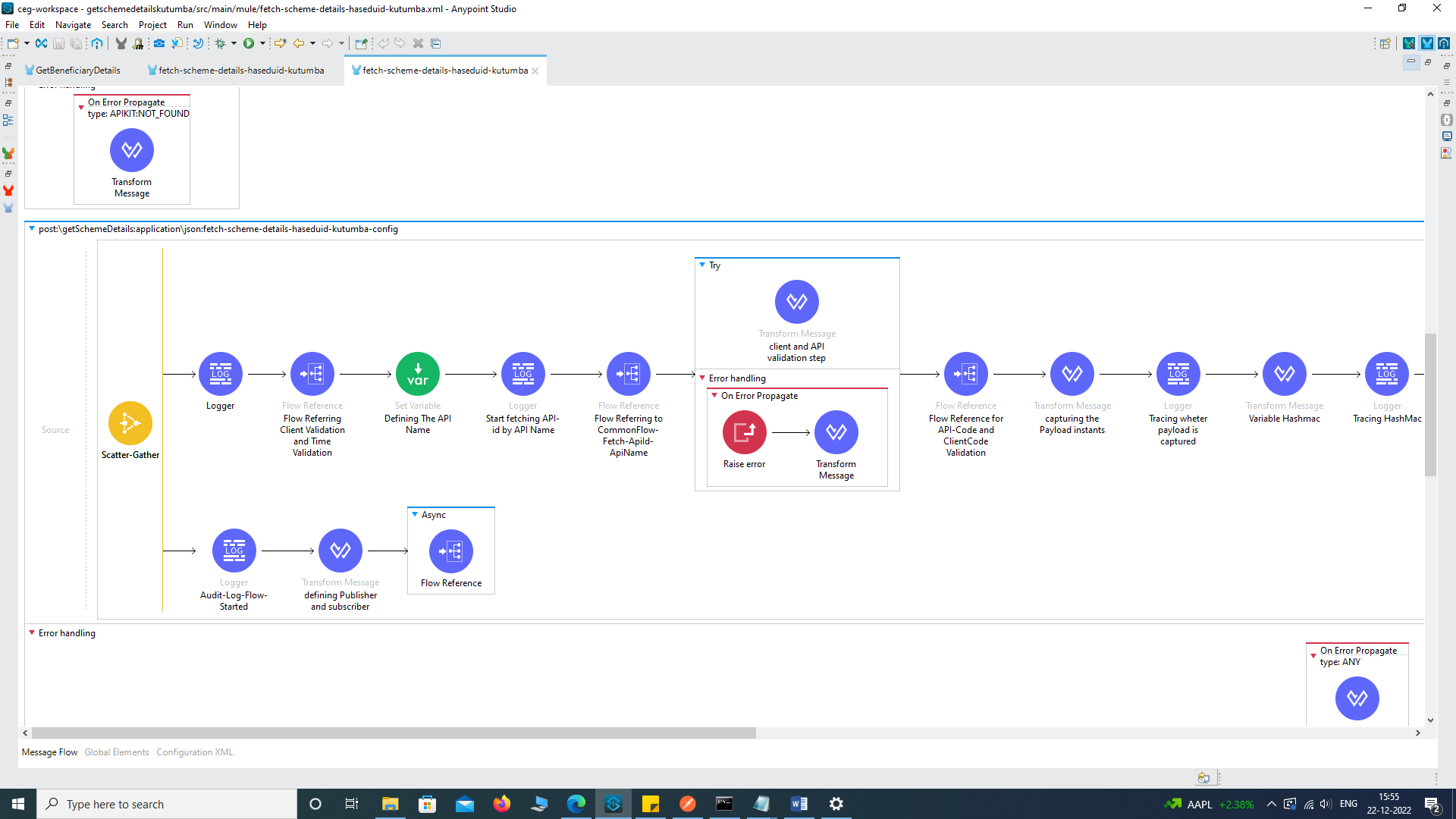
Effective Error handling is done at each level of validation process:refer the APi's implemented

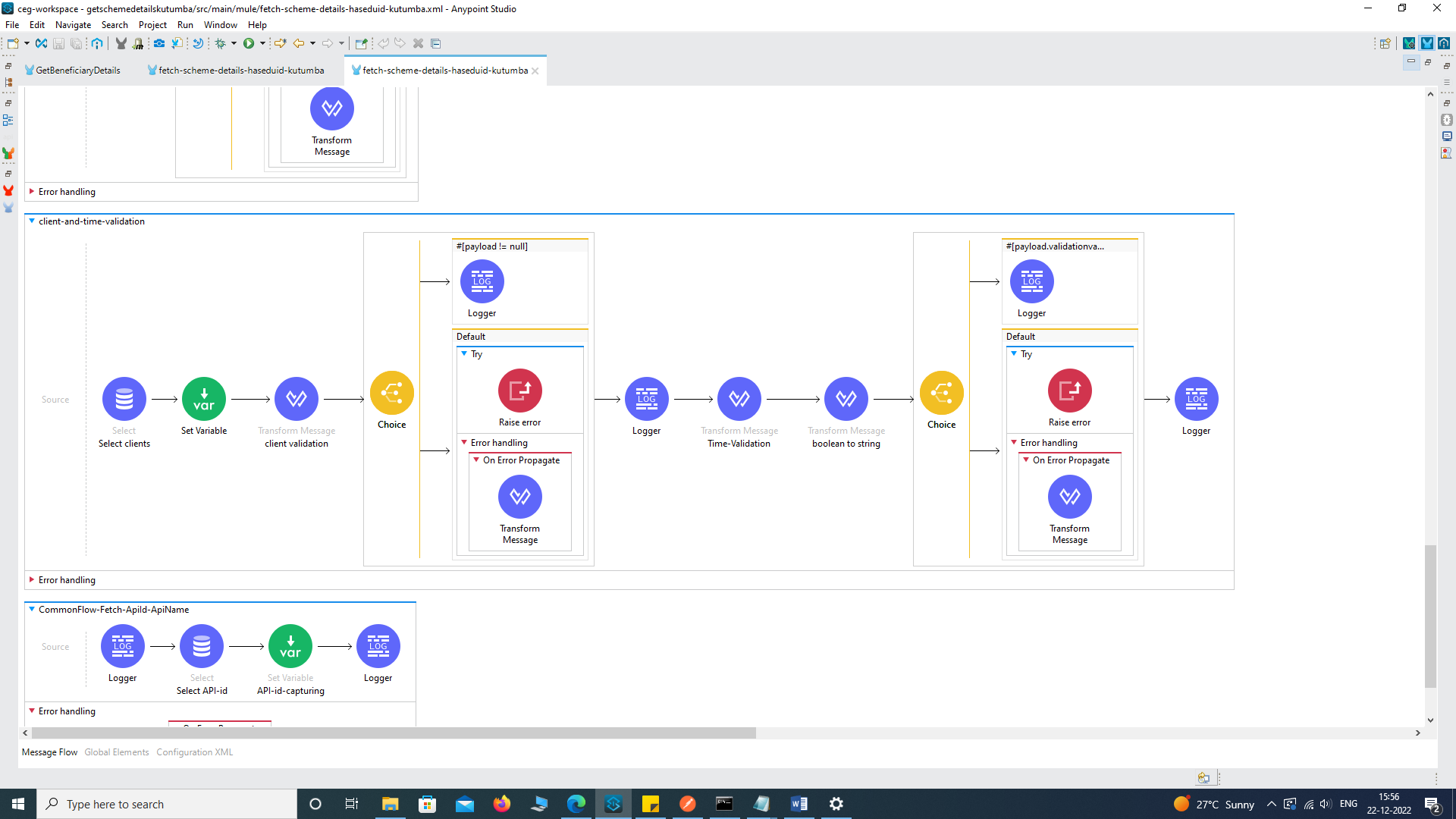
Now entered clients Alone can make use of the Api's effectively and also we can restrict the Access to particular client whenever necessary.

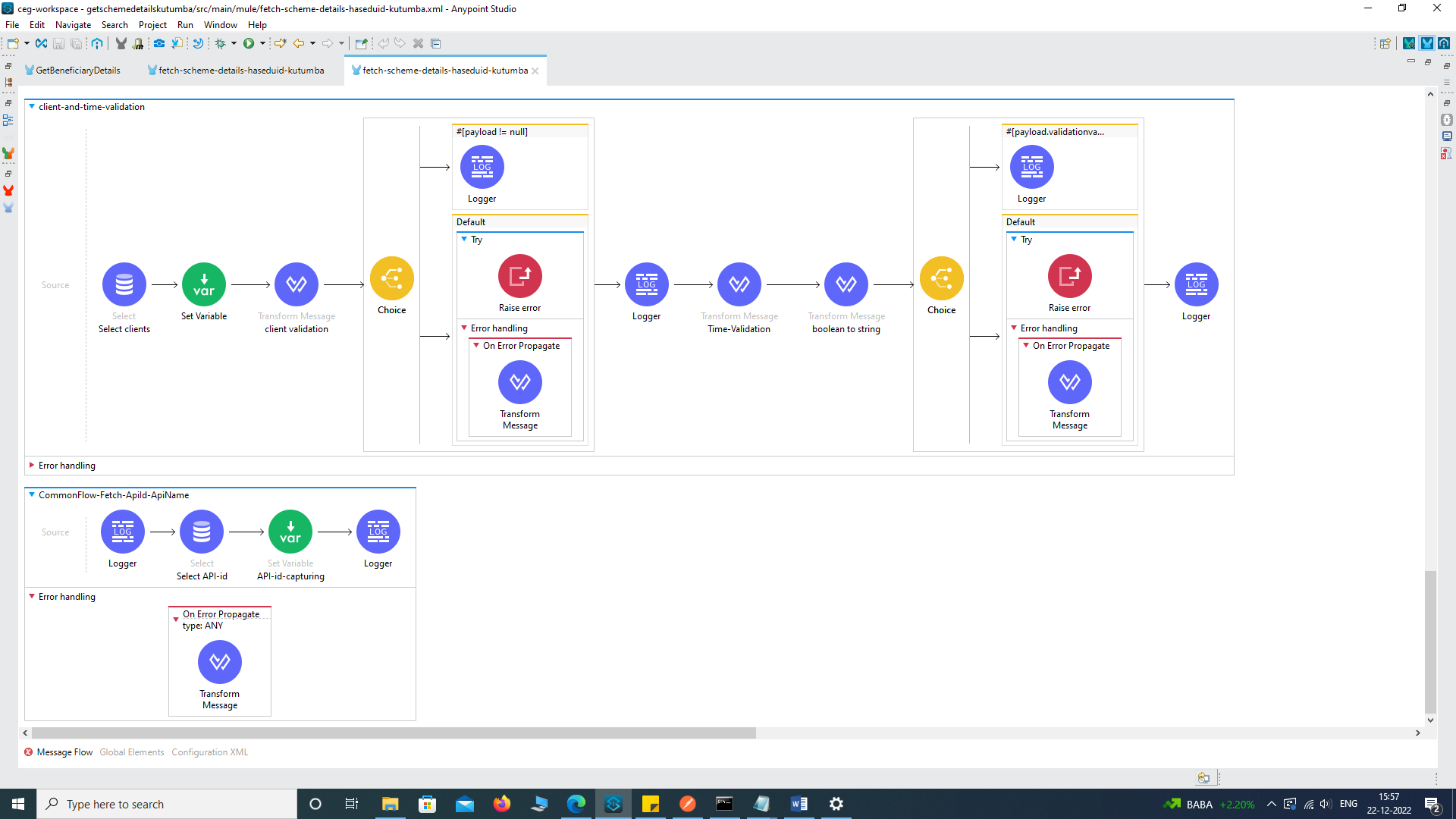
Refer this File for the Future use case.

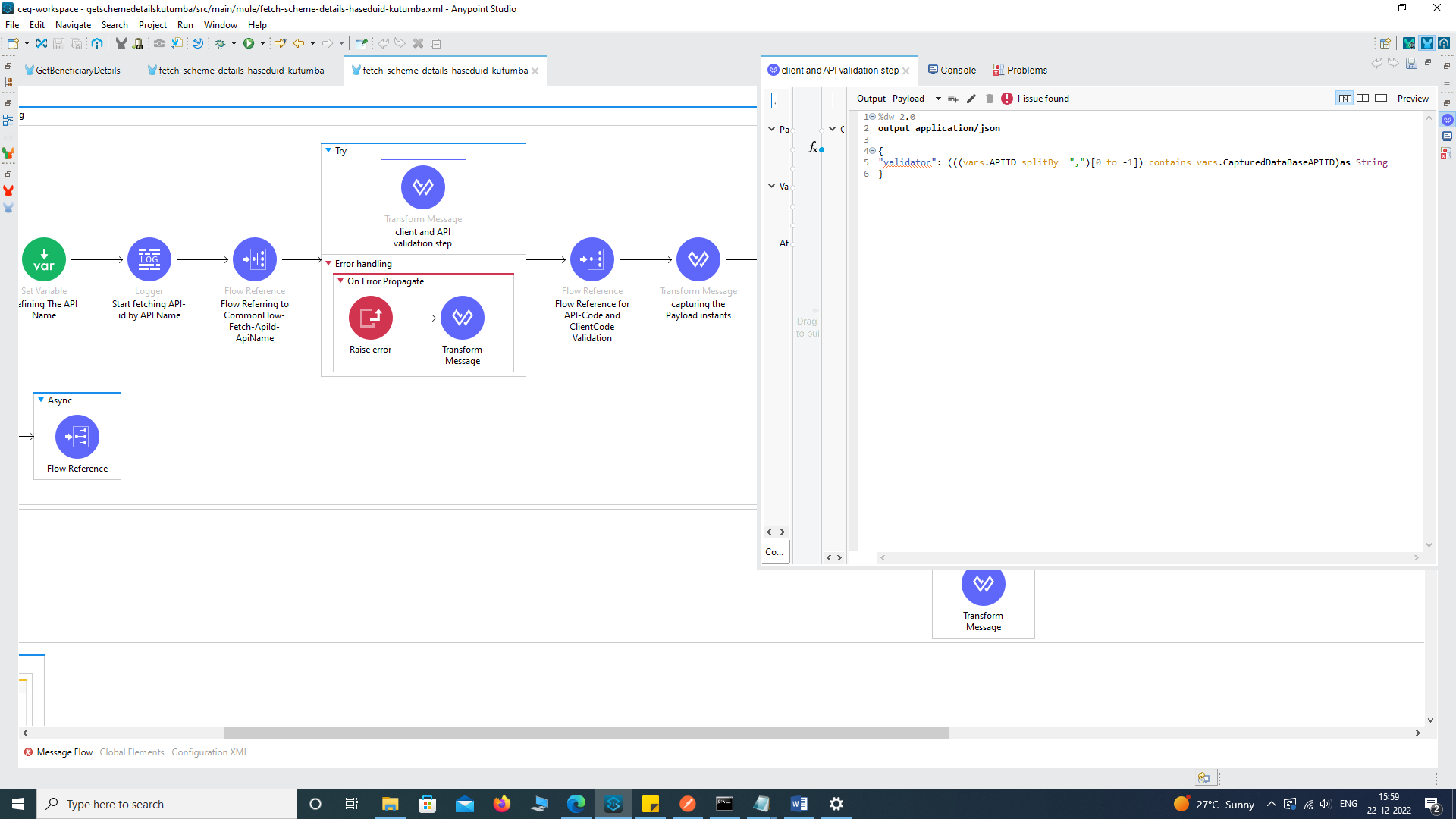
--------------------------------------------------------------------------------------------------------------------------

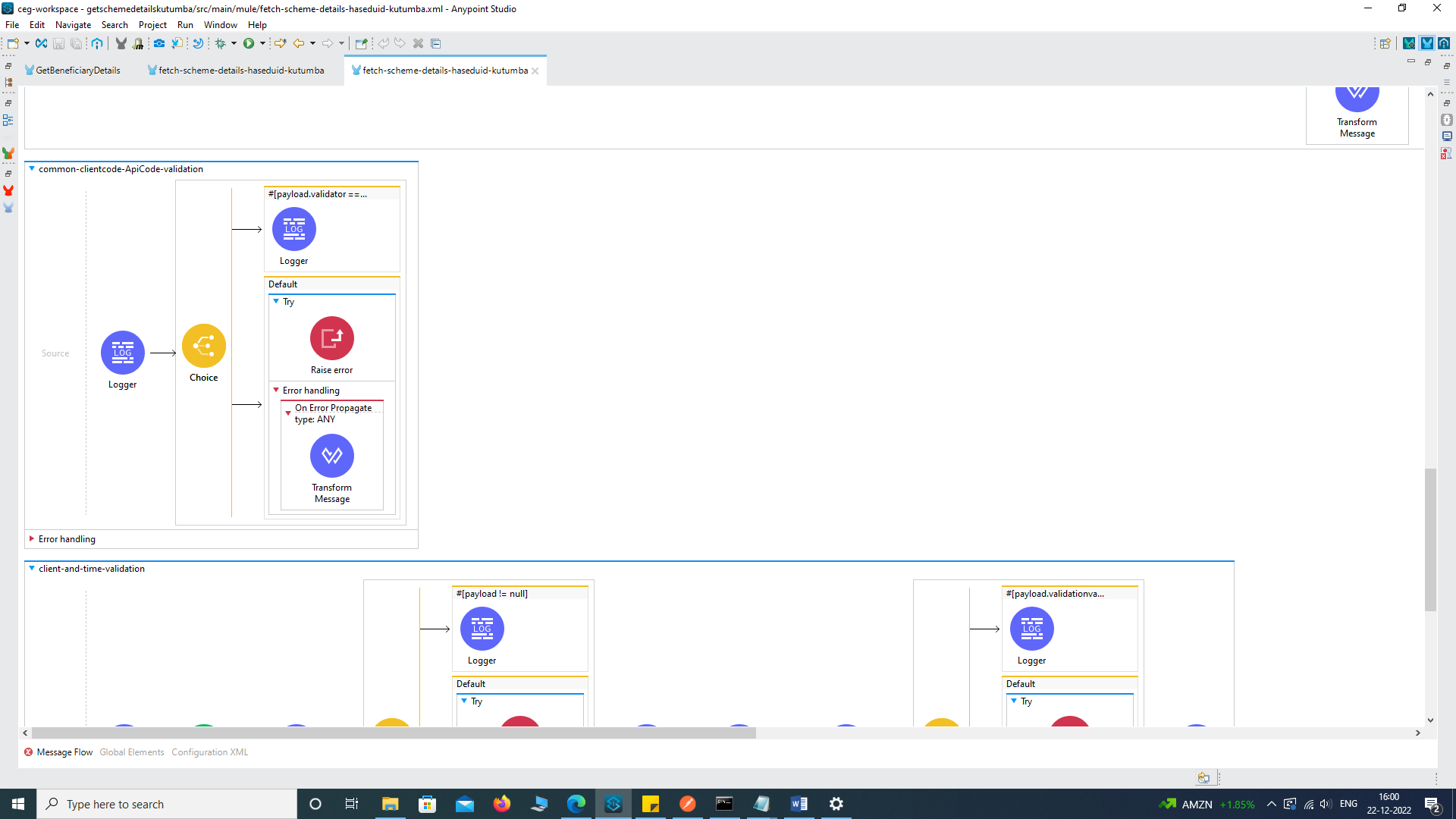


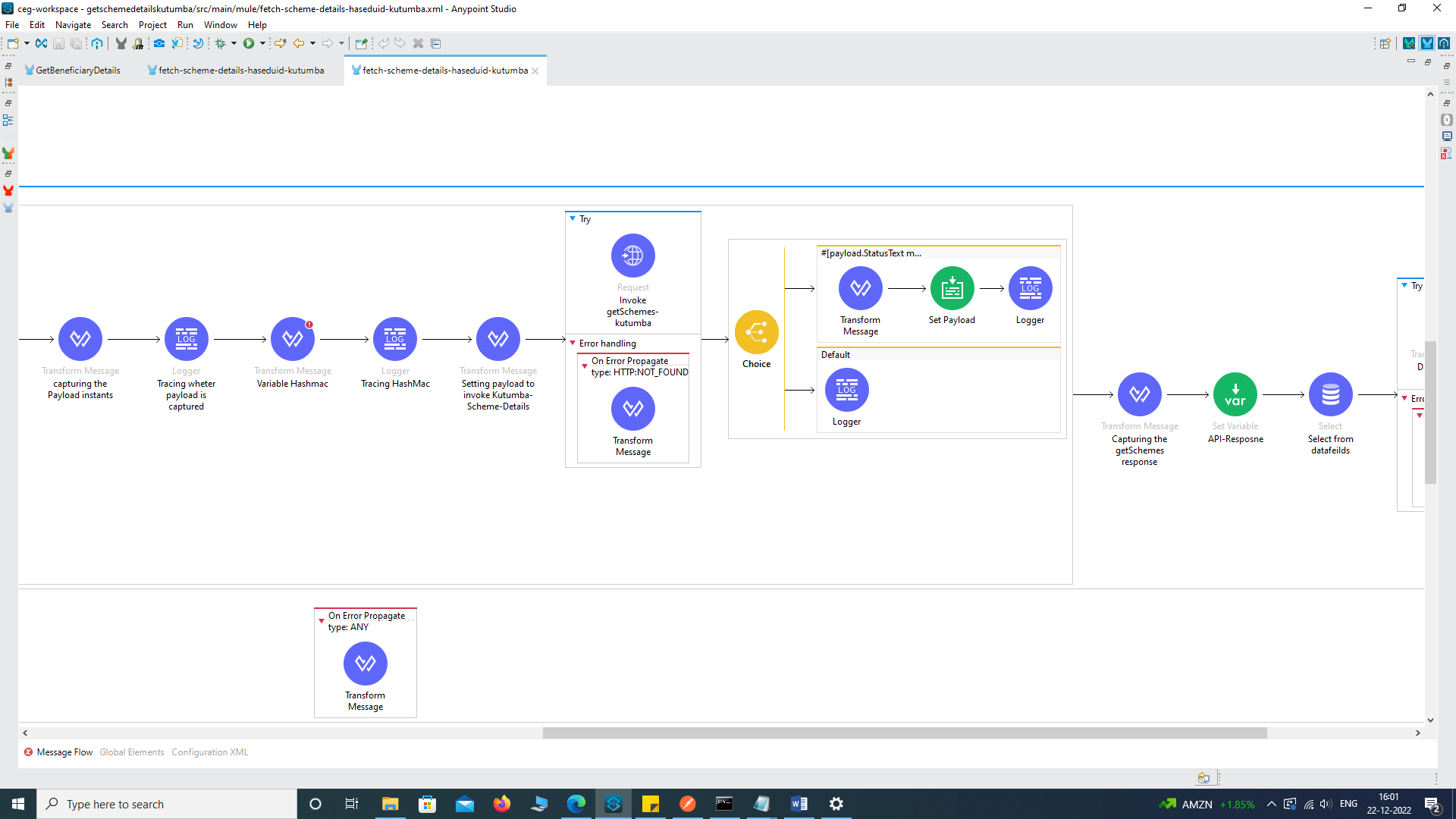


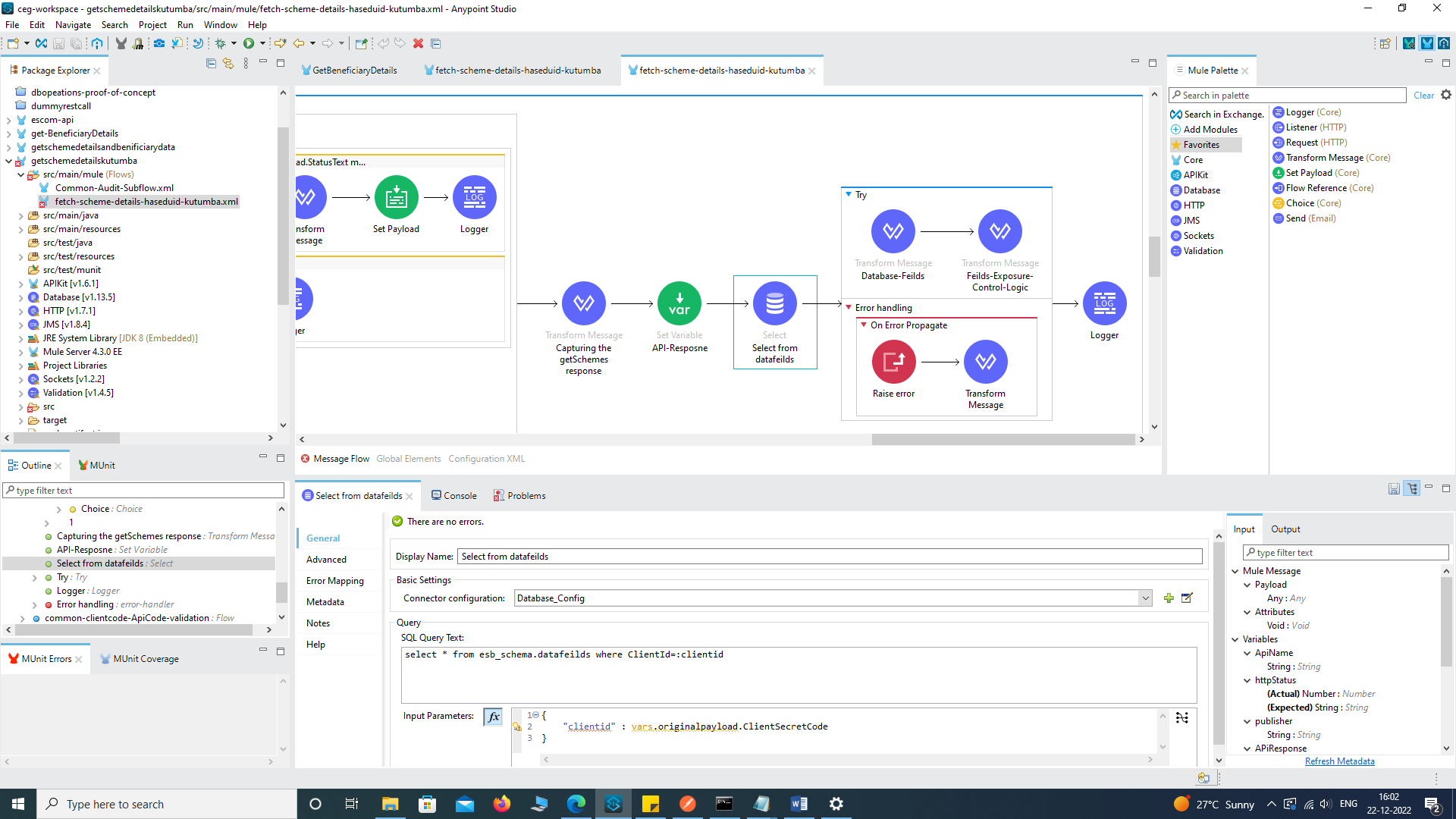


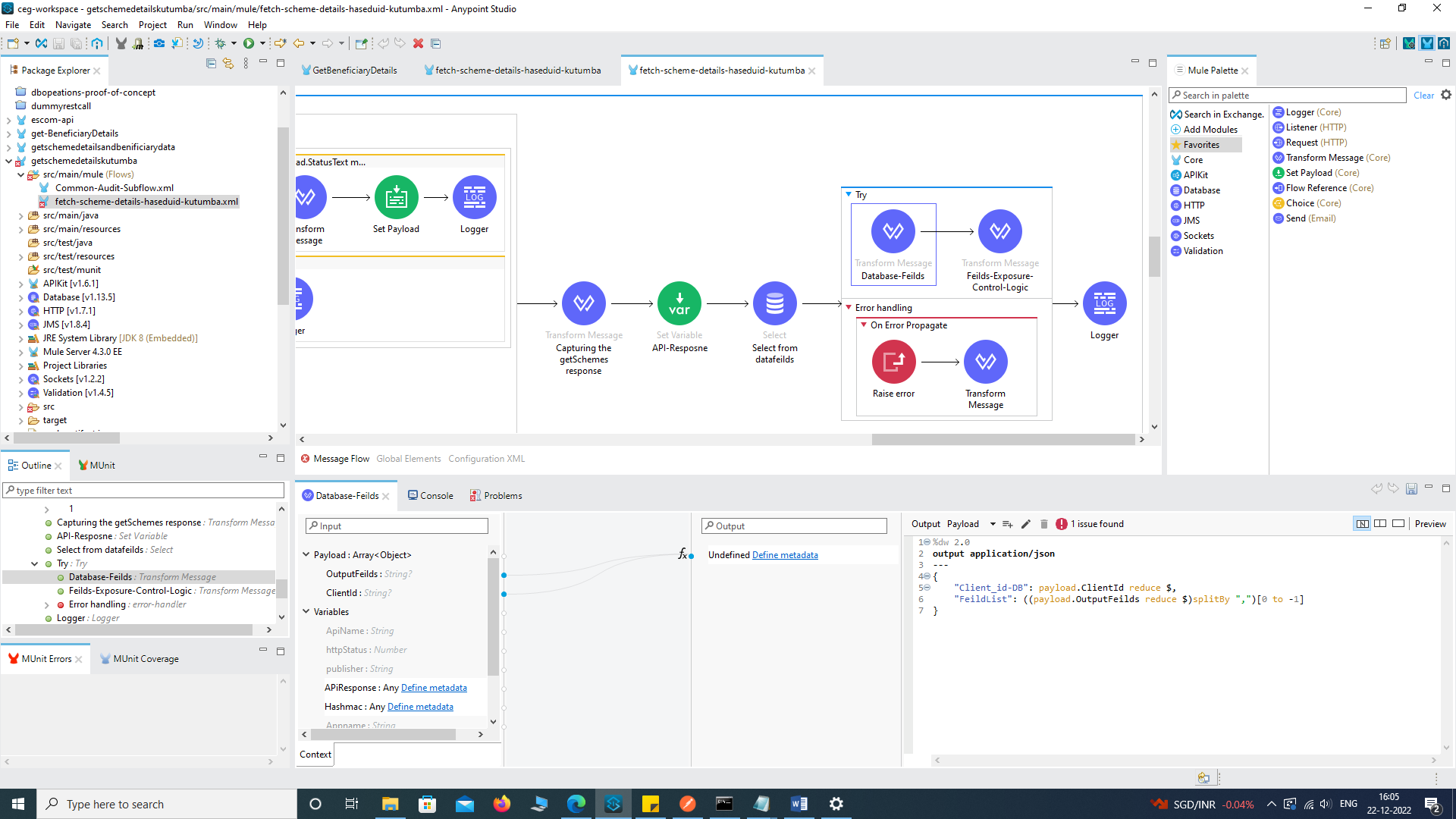


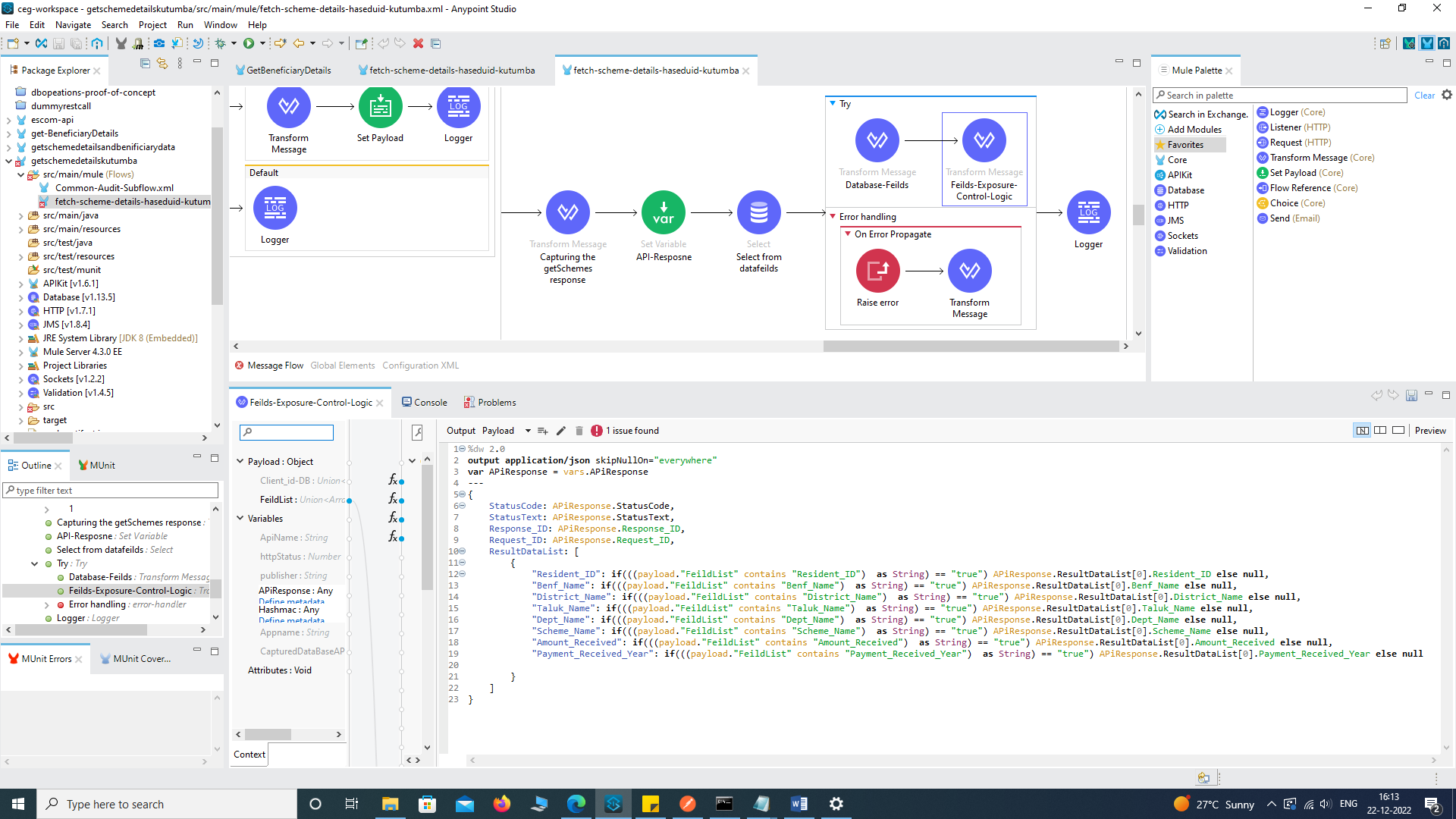












END OF THE DOCUMENT