**MuleSoft Assignment**

Table of Contents

[Part 1- Troubleshoot Exercise 3](#_Toc102916600)

[Troubleshoot 3](#_Toc102916601)

[How to troubleshoot the application 3](#_Toc102916602)

[Error on Console: 3](#_Toc102916603)

[Troubleshooting steps 3](#_Toc102916604)

[Run Memory Analyser Tool: 4](#_Toc102916605)

[Memory Leakage: 5](#_Toc102916606)

[Analysis 7](#_Toc102916607)

[Resolution 7](#_Toc102916608)

[Part 2- Use case discussion 10](#_Toc102916609)

[Introduction: 10](#_Toc102916610)

[Vision: 10](#_Toc102916611)

[How we can contribute to their success 10](#_Toc102916612)

[Flow Diagram 11](#_Toc102916613)

[Technical Usecase: 11](#_Toc102916614)

[API Led Integration: 12](#_Toc102916615)

[Microservices Developed 12](#_Toc102916616)

[**Third Party system Integrated in Mule Application** 12](#_Toc102916617)

[Details of Mule Application 13](#_Toc102916618)

[Pre-Requisite to run the Mule Application 22](#_Toc102916619)

[Attachment: 22](#_Toc102916620)

[Postman collection: 22](#_Toc102916621)

# Part 1- Troubleshoot Exercise

## Troubleshoot

### How to troubleshoot the application

* Run the Mule Application

### Error on Console:

# java.lang.OutOfMemoryError: Java heap space # - XX:OnOutOfMemoryError="taskkill /F /PID %p“ # Executing "taskkill /F /PID 24712"... JVM exited unexpectedly. Automatic JVM Restarts disabled. Shutting down

**Initial Analysis**

* + Above error clearly states that JVM got disabled of memory Out of exception. This error occurs because of Heap memory was exhausted

### Troubleshooting steps

1. Configure the path in wrapper.config file to create the heap dumps

wrapper.java.additional.19=-XX:HeapDumpPath= C:\MuleSoft\AnypointStudio-7.12.0-win64\AnypointStudio\plugins\org.mule.tooling.server.4.4.0.ee\_7.11.0.202203290350\mule\logs

2. Run the mule application again.

3. Now the heap dump is generated when the application again fails.

**Text, letter

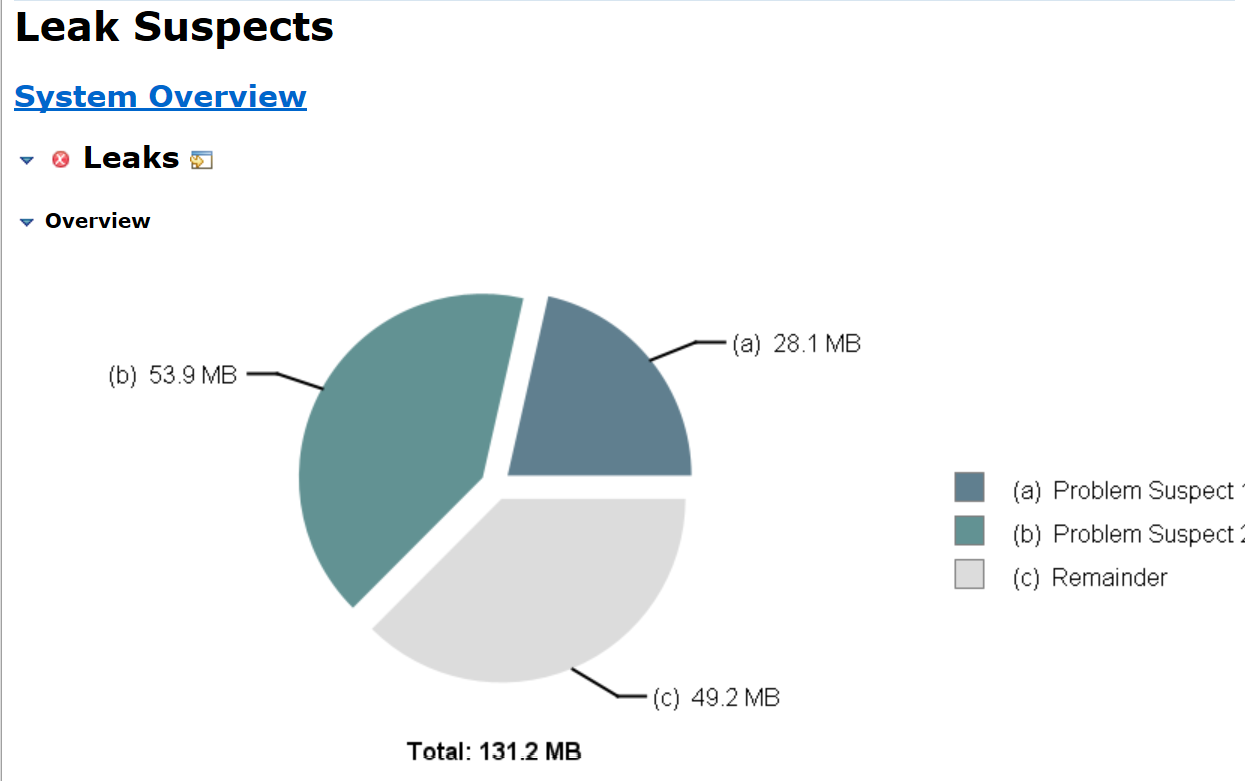
Description automatically generated**

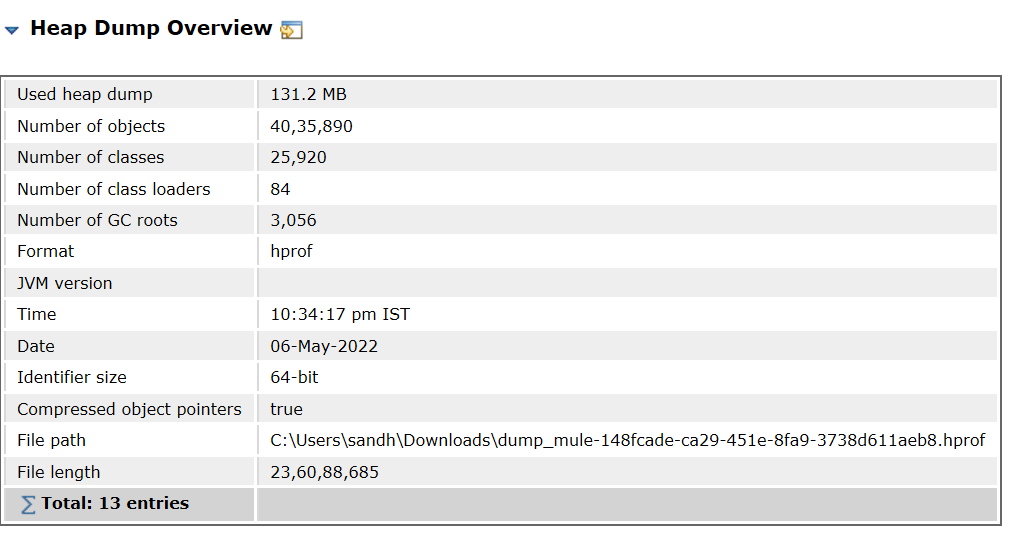
### Run Memory Analyser Tool:

1. Collect the heap dumps HPROF file and open the file using the Memory Analyzer tool.
2. Issues highlighted was Memory leakage.

Graphical user interface, application

Description automatically generated





### Memory Leakage:

**Suspect Issue1:**

One instance of **org.mule.runtime.core.internal.util.queue.TransactionalQueueManager** loaded by **org.mule.runtime.module.reboot.internal.MuleContainerSystemClassLoader @ 0xf6e07f60** occupies **2,94,96,552 (21.44%)** bytes. The memory is accumulated in one instance of **java.util.LinkedList**, loaded by **<system class loader>**, which occupies **2,94,94,224 (21.44%)** bytes.

Keywords:

**Keywords**

* org.mule.runtime.core.internal.util.queue.TransactionalQueueManager
* org.mule.runtime.module.reboot.internal.MuleContainerSystemClassLoader @ 0xf6e07f60
* java.util.LinkedList

Verified the stack trace:

[**Thread Stack and Involved Local Variables**](file:///C:\Users\sandh\AppData\Local\Temp\report12548777425652763663\pages\27.html#content)

**Suspect Issue 2:**

48 instances of **java.lang.Thread**, loaded by **<system class loader>** occupy **5,65,11,888 (41.08%)** bytes.’

Detail description:

Biggest instances:

* java.lang.Thread @ 0xf7ce2ce0 [MuleRuntime].uber.08: [troubleshooting-exercise].troubleshooting-exerciseFlow.CPU\_LITE @8472f2d - 1,25,57,216 (9.13%) bytes.
* java.lang.Thread @ 0xf7ce8d38 [MuleRuntime].uber.10: [troubleshooting-exercise].troubleshooting-exerciseFlow.CPU\_LITE @8472f2d - 81,07,752 (5.89%) bytes.
* java.lang.Thread @ 0xf7ce27f0 [MuleRuntime].uber.06: [troubleshooting-exercise].troubleshooting-exerciseFlow.CPU\_LITE @8472f2d - 56,40,144 (4.10%) bytes.
* java.lang.Thread @ 0xf7ce2578 [MuleRuntime].uber.05: [troubleshooting-exercise].troubleshooting-exerciseFlow.CPU\_LITE @8472f2d - 55,74,080 (4.05%) bytes.
* java.lang.Thread @ 0xf7ce2088 [MuleRuntime].uber.01: [troubleshooting-exercise].troubleshooting-exerciseFlow.CPU\_LITE @8472f2d - 53,05,088 (3.86%) bytes.
* java.lang.Thread @ 0xf7ce2300 [MuleRuntime].uber.04: [troubleshooting-exercise].troubleshooting-exerciseFlow.CPU\_LITE @8472f2d - 45,97,064 (3.34%) bytes.
* java.lang.Thread @ 0xf7ce1e10 [MuleRuntime].uber.02: [troubleshooting-exercise].troubleshooting-exerciseFlow.CPU\_LITE @8472f2d - 39,43,872 (2.87%) bytes.
* java.lang.Thread @ 0xf7ce8ac0 [MuleRuntime].uber.09: [troubleshooting-exercise].troubleshooting-exerciseFlow.CPU\_LITE @8472f2d - 34,65,224 (2.52%) bytes.
* java.lang.Thread @ 0xf7ce1b98 [MuleRuntime].uber.03: [troubleshooting-exercise].troubleshooting-exerciseFlow.CPU\_LITE @8472f2d - 29,57,080 (2.15%) bytes.
* java.lang.Thread @ 0xf7ce1690 [MuleRuntime].uber.12: [troubleshooting-exercise].troubleshooting-exerciseFlow.CPU\_LITE @8472f2d - 22,37,008 (1.63%) bytes.
* java.lang.Thread @ 0xf7ce1920 [MuleRuntime].uber.11: [troubleshooting-exercise].uber@org.mule.runtime.core.internal.source.scheduler.DefaultSchedulerMessageSource.createScheduler:227 @429696ef: [troubleshooting-exercise].troubleshooting-exerciseFlow.CPU\_LITE @8472f2d - 20,66,032 (1.50%) bytes.

**Keywords**

* java.lang.Thread

### Analysis

1. Memory Analyzer tool logs its clearly states that this issue is because of memory leakage.
2. Analyzed the mule Application can could see there are 2 schedulers running.
   * + Scheduler 1: Publishing the message to the VM queue in every 1000 ms.
     + Scheduler 2: Subscribing the message in every 7000 ms.
     + Threads were getting to pill up as publishing rate was greater than consuming rate.

### Resolution

* Make the changes in the mule Application so that consumer can publish the message as soon as the publisher publishes the message
* In this case decrease the scheduler timing from 7000 ms to 1000 ms and re run the process. Mule App started running without any out of memory Exception.
* Another way to handle this is to increase the VM memory argument to some higher value so that whatever the instances are getting generated heap memory should be able to handle it.
* In this case as there was no resource crunch, Reverted all the changes done in Mule App and change the VM arguments with below value –Xmx512m -Xms512m.
* Run the mule App again. With higher memory allocated to the heap it worked smoothly

# Part 2- Use case discussion

Introduction:Acme International is customer known for providing energy and utilities services using the digital platform.

Vision:Acme International wants to make people life easier by introducing the technology to their lives. Acme International wants to provide the platform to the customer where the customer can signup and use the available services.

How we can contribute to their success**:** Acme international wants to integrate their system with different other system using our integration product **“MuleSoft”.** Customer wants to use best technology to integrate their system and provide seamless experience to their customers.

### Flow Diagram

Diagram

Description automatically generated

Technical Usecase: Below table explains what business requirements are and to cater them what all technical use UseCases come for development activity

|  |  |
| --- | --- |
| **Business Use case** | **Technical use case** |
| Self-service portal | Using Salesforce CRM to register the customer details |
| View Bills | Using Salesforce and MongoDB REST APIs to get the details |
| Pay Bills | Integrating with third-party payment gateway APIs |
| Download bills | Using Salesforce, MongoDB APIs and transform using Data weave |
| Handling error | Global error handler to handle all kinds of error |
| Subscribe alert for notification | Using Messaging features |

API Led Integration:With above description in the table it provides the details what all API services needs to be developed for achieving the business requirements.

### Microservices Developed

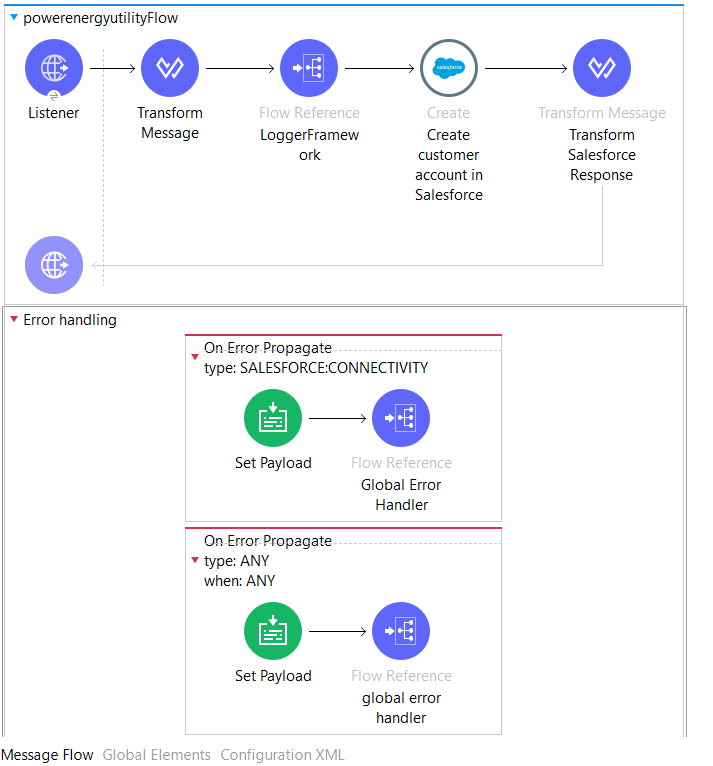
* **Real Time API:** 
  + API exposed to customer for signup process
  + API exposed to update the personal details of customer
  + API exposed to check the tariff plans for their utilities.
  + API exposed to generate the bill on Demand
  + API exposed to pay the Bill
* **Batch processing**
  + Monthly generate the customer Bill
  + Sending email notification to customer when the bill is generated
* **Publish subscribe:**
  + Customer Subscribe Newsletter
* **Reusable Framework:**
  + Common Logger framework
  + Global error handler

### **Third Party system Integrated in Mule Application**

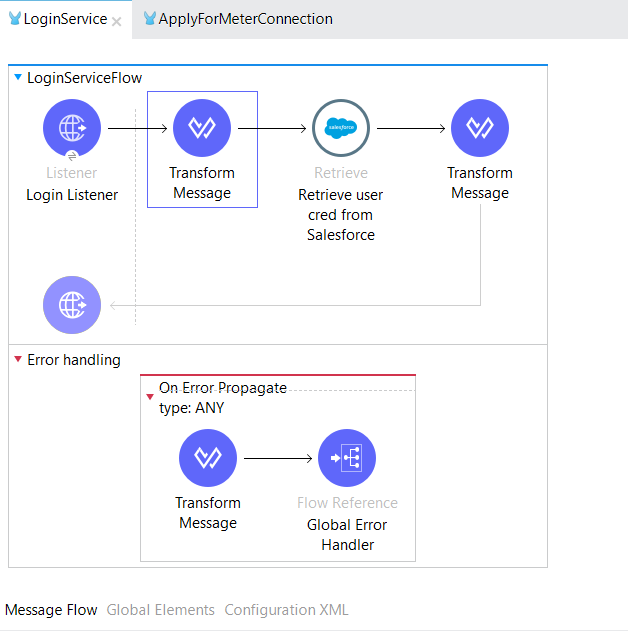
* + Salesforce
  + MongoDB
  + Azure Service Bus
  + Azure Storage
  + Third Party payment Gateway system

### Details of Mule Application

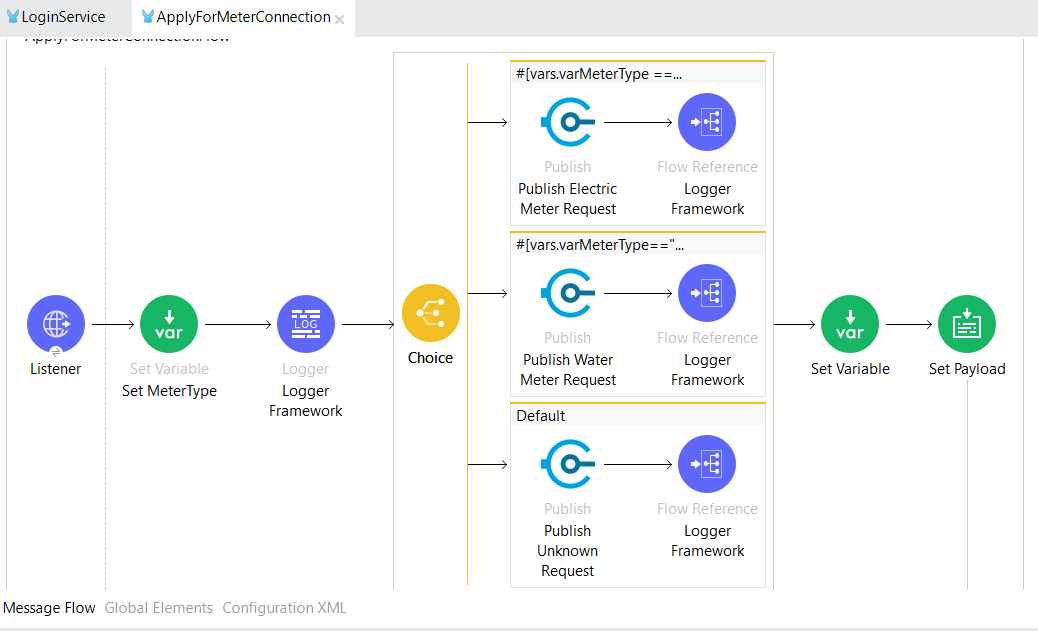
**Signup Process:** For sign up on the portal, **portaleneryutility** service is created. This API is exposed with http end points.



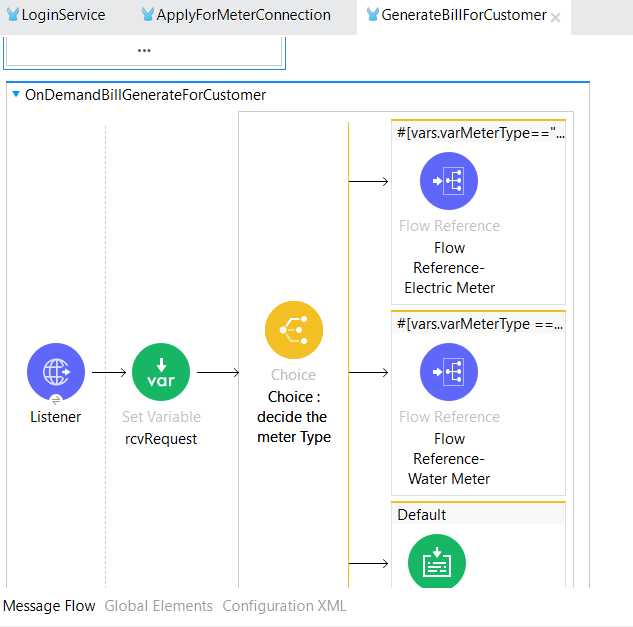
**Login Service:** This service is used to validate the user credentials available in Salesforce and once authenticated customer can see the other details on the portal. To do same **LoginService** is exposed as an API with http points.

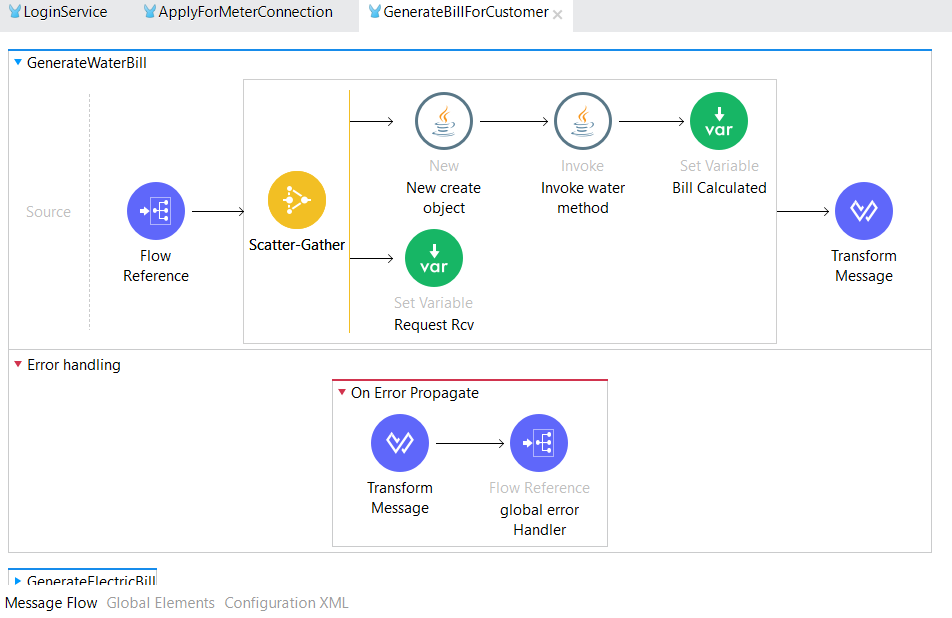
****

**Apply for Utility meter:** For applying the meter connection from the portal, **ApplyForMeterConnection** service is created. This API is exposed with http end points.

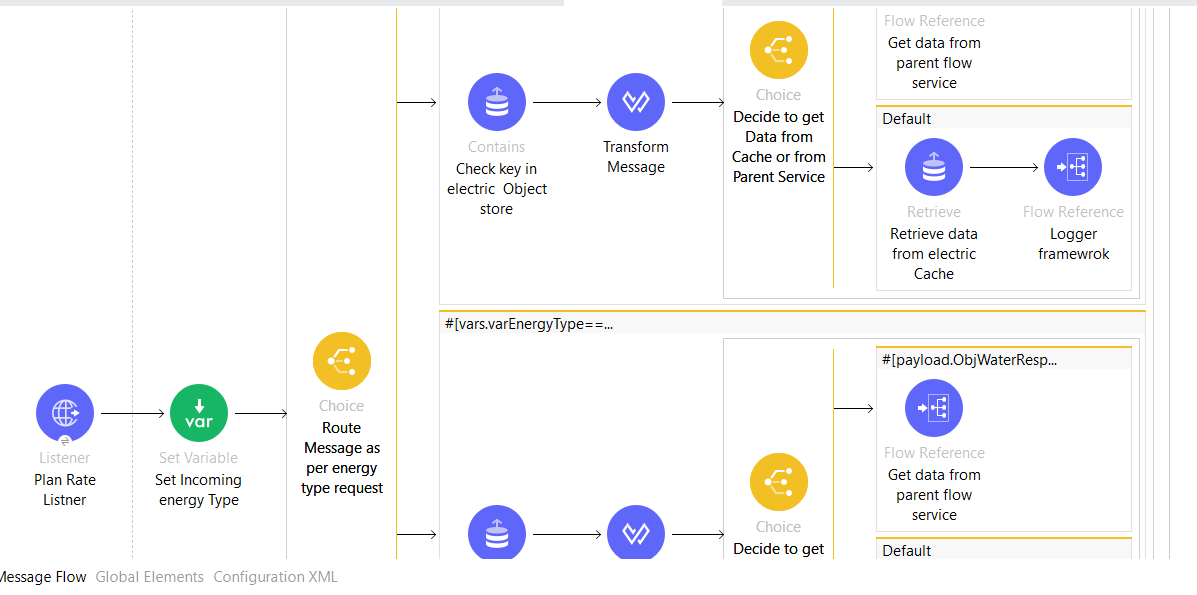


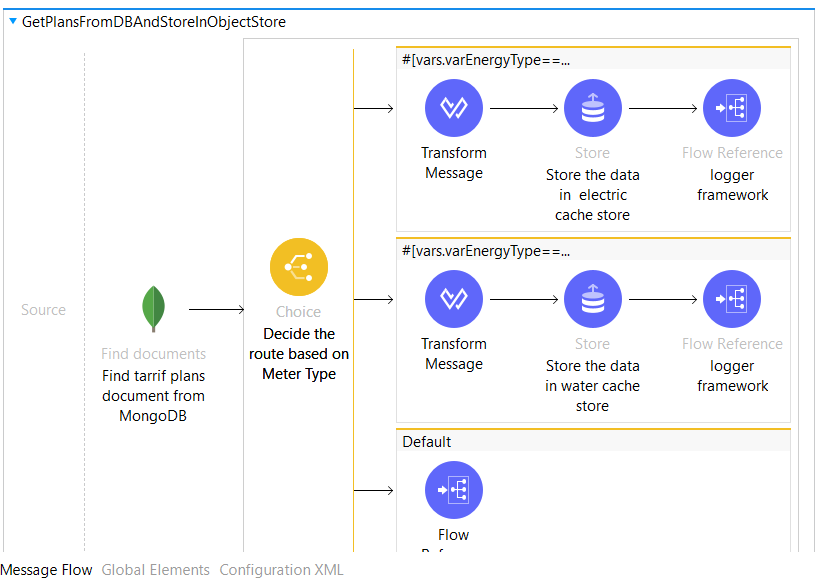
**Generate Bill:** Customer can generate their bill on demand using **GenerateBillforCustomer** API exposed.

****

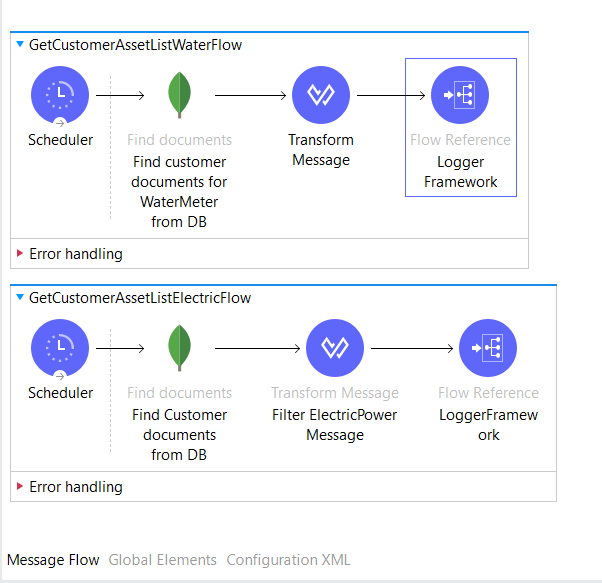
****

**Get Tariff Details:** Customer can get the details of different plan rates using the **EnergyPlanRates.** These plans provide the details for residential and commercial usage.

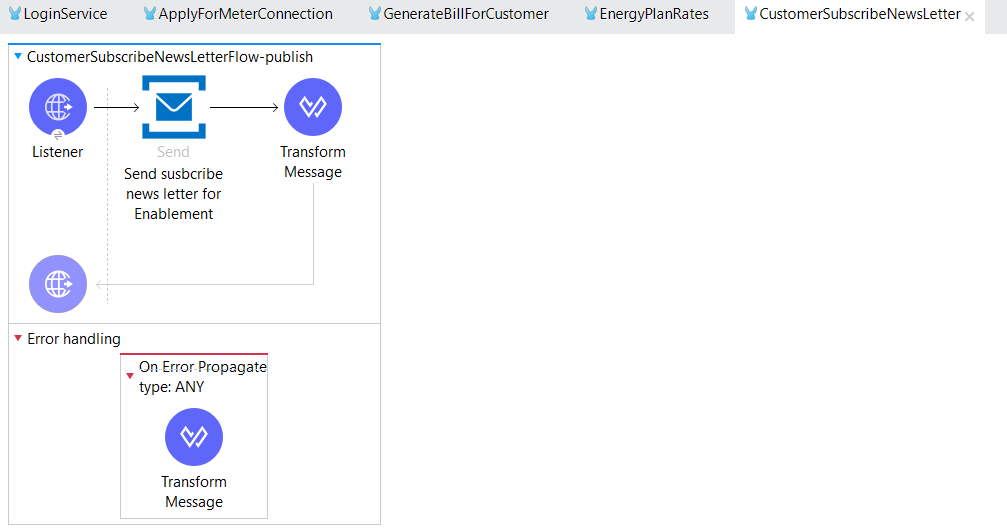


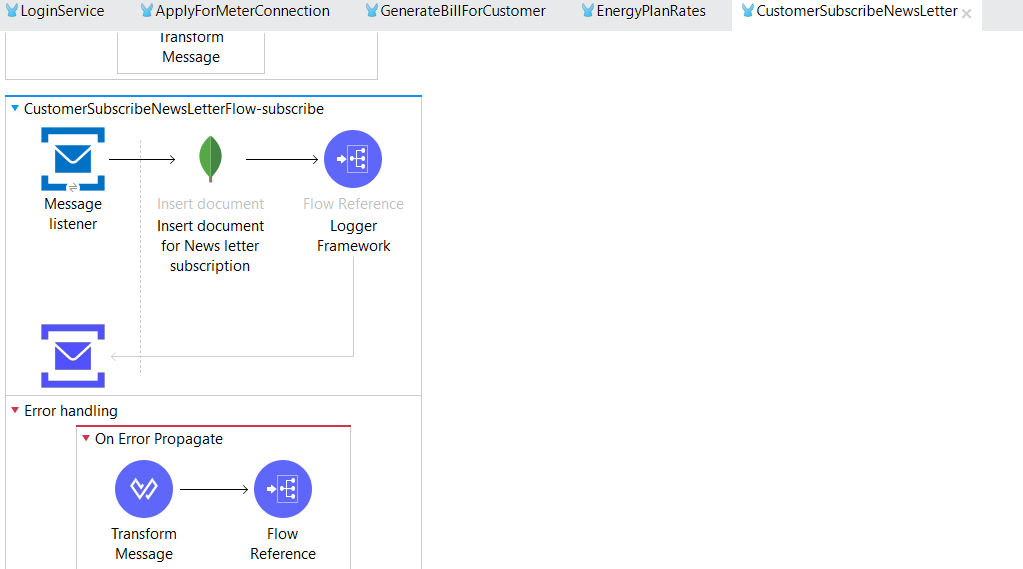


**GetCustomerAsset List**: This is scheduler service which pulls the data from MongoDB. This service provides the details of customer and all their meter details. Separate schedulers are created to pull the data from MongoDB and load this data into Azure Storage. This Data will be used for Analytics purposes.

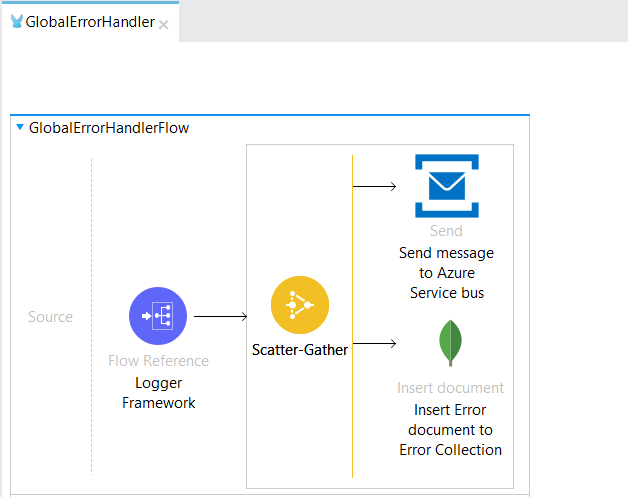


**Subscribe Newsletter:** Customer can subscribe for the newsletter. To do same **CustomerSubscribeNewsLetter** is created and exposed as an API endpoints.





**Global Error Handler:** This is global service framework created to handle all kinds of error occurred in the mule Application while running any of the microservices. TO do same **GlobalErrorHandler** API is created in the muleApplication.

****

**Logger Framework:** This is global service framework created to log all kinds of payload in the mule Application while running any of the microservices. TO do same **LoggerFramework** API is created in the muleApplication.

Graphical user interface, application

Description automatically generated

**Scheduler To generate Bill monthly**: This is scheduled service run monthly to generate the bill and update customer using email notification. To do same **schedulerToGenrateBillAndNotificationToCustomerFlow** is created as scheduler.

Graphical user interface, application, Word

Description automatically generated

## Pre-Requisite to run the Mule Application

* 1. Running instance of MongoDB
  2. Salesforce CRM account
  3. Azure subscription account for Azure service bus
  4. Azure subscription account for Azure storage services

### Attachment:

* 1. **Mule Application jar file:** 
     + Install the jar file on AnyPoint studio.

****

### Postman collection:

* + - [**https://www.getpostman.com/collections/ec8e616fcfd779564f43**](https://www.getpostman.com/collections/ec8e616fcfd779564f43)