**DOCUMENTATION FOR PLHG-ITEMS-PRODUCT-PRICEBOOK**

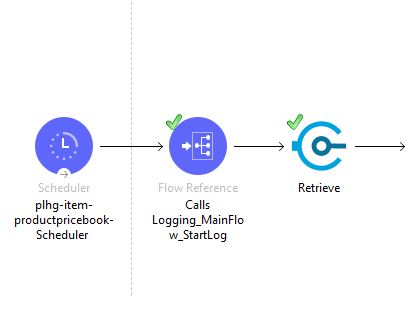
**Logging Framework:**

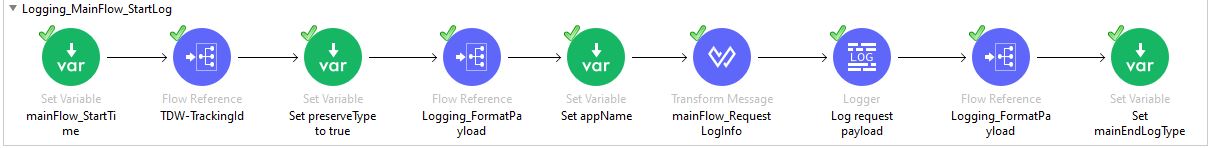
**Logging is useful for monitoring and troubleshooting your Mule applications and the Mule server — whether that's recording errors raised by the application or specific details, such as status notifications, custom metadata, and payloads**

**In this we have used common-logging.xml, common-error-handlers.xml, trackingid.xml to apply the logging framework for the flow**

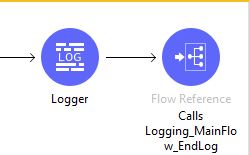
**Common-logging.xml:**

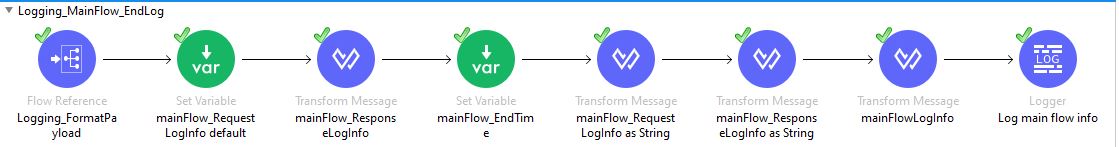
**In Endpoints flow, we used flow reference at the start of the flow to call the main-flow-startlog in the common-logging.xml file. Whenever the flow starts the flow reference components triggers the start-log flow.**

****

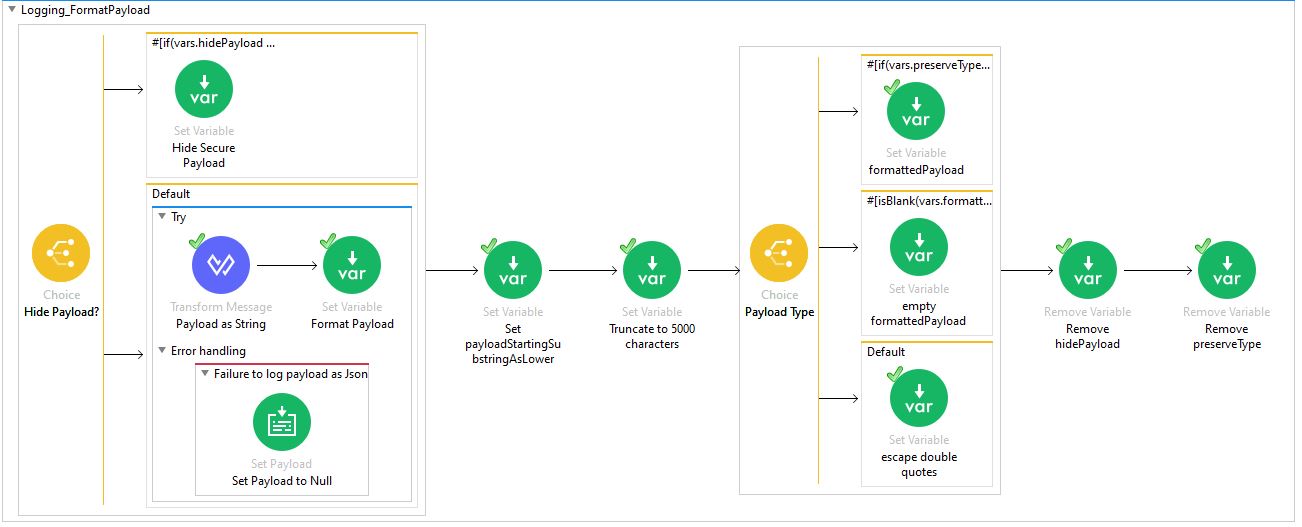
****

**In Endpoints flow, we used another flow reference component to call the main-flow-endlog in the common-loggin.xml file. This flow reference component triggers the End-log flow**

****

****

**In both Logging-main flow-startLog and logging main flow -endLog there is a flow reference component which calls the Logging-format payload**

****

**M-Unit Test cases for the flow:**

**MuleSoft has a framework called MUnit that allows us to write automated test cases for our APIs and integrations. It’s a perfect fit for a continuous integration/deployment environment.**

**M-Unit Use Cases:**

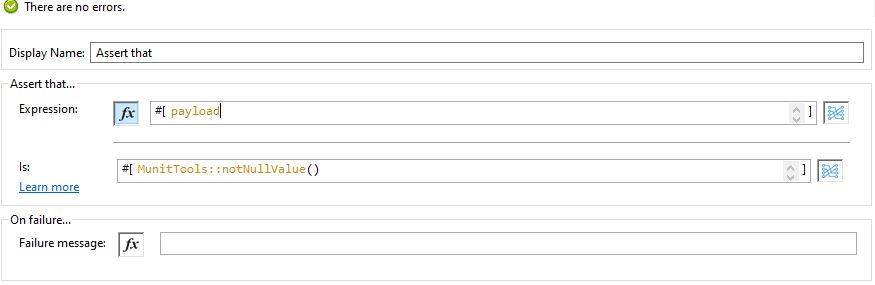
* **Check visual coverage in the studio**
* **Verify message processor calls**
* **Spy any component in the flow**
* **Mock outbound endpoints**
* **Mock message processors**
* **Enable or disable particular tests**
* **Debug your tests with studio**

**To create a specific test for the flow, Right click on the flow, then select M-unit and click on Create new Suite. It will create a test for you in src/test/munit folder**

**M-Unit Components:**

**Assert That:**

**The Assert That event processor allows you to run assertions to validate the state of a Mule event’s content. You can use the Assert That processor to validate the Mule event after the production code runs**

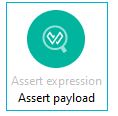
****

**The Assert That processor uses a set of DataWeave functions called MUnit matchers to define the assertion conditions for any value in an expression.**

**We can check if the payload is null using nullValue( ) matcher. If the Assertion fails, then the processor will throw java.lang.AssertionError.**

**Assert Expression:**

**The Assert Expression processor allows you evaluate a Dataweave expression that asserts the state of a Mule Event’s content. This is used to validate the mule event after the production code runs**

****

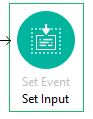
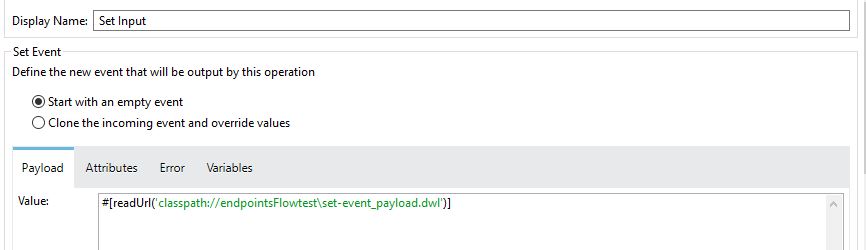
**Fail:**

**The Fail Event Processor allows you to fail your test on purpose. This is useful to validate that a test should fail if that point is reached. The message field is optional and defines the error message to print when the assertion fails. The processor throws a java.lang.AssertionError.**

**Set Event:**

**The Set event processor allows you to define a mule event. The Set event processor is used at the beginning of M-unit test, to define the first to send to the flow being tested.**

**The set-event has the cloneOriginalEvent property. If set to true, it clones the event produced by your code. By default, the value of this property is false.**

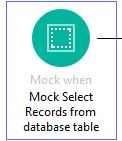
** **

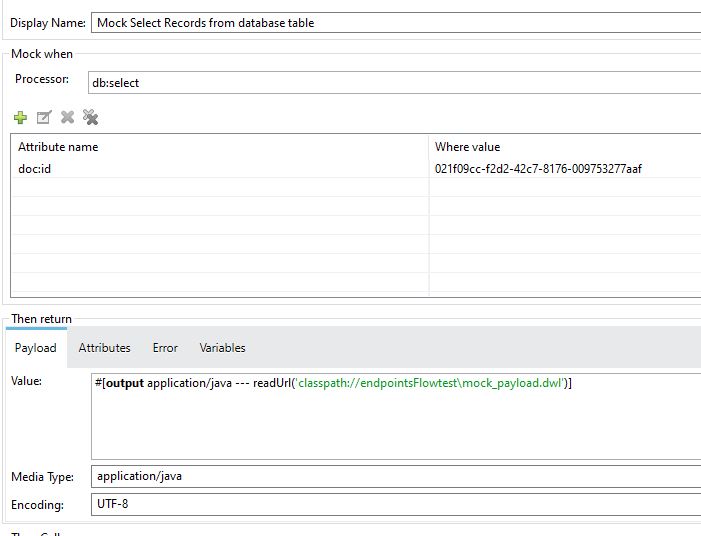
**Attributes in Set Event:**

* **Value: Defines the value of payload message**
* **Encoding: It is optional attribute that defines the encoding of message**
* **MediaType: It is optional attribute that defines the media type of message**

**Mock When:**

**The Mock When processor allows you to mock an event processor when it matches the defined name and attributes. In this flow we mock the database connector and Salesforce connector.**

****

****

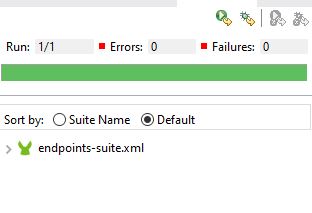
## **Verify Call:**

**This processor is defined to verify if any processor was called. Here we can validate if a specific message processor has been called by using a set of attributes with a specific number of times.**

* **Processor: Describes which event processor you want to mock. The description takes the form {name-space}:{event-processor-name}. It supports regular expressions.**
* **Times: It defines the verification as successful if the event processor was called *N* number of times.**
* **At least: Defines the verification as successful if the event processor was called a minimum of *N* number of times.**
* **At most: Defines the verification as successful if the event processor was called a maximum of *N* number of times.**

**How to run m-unit test:**

**Right click on m-unit test and run m-unit test. Once the munit flow is triggered, it will provide details such as errors, failures, coverage report and test status.**

****

**In the above figure, the green color line indicates that the flow is successfully executed and passed all test cases. After running the munit flow it also generates the report with overall coverage.**

**When if the munit test fails, the generated error report displays as red color**

**The Coverage report provides coverage of how successfully the mule application has been executed by a set of munit tests**