**Mule Flow and Asynch**

**Mule Flow:**

\* Mule apps process messages and other parts of Mule events through Mule components, connectors, and modules that are set up within the scope of Flow and Subflow components within an app.

\* Flows provide the most robust and flexible way to build Mule applications, because you can arrange convenient building blocks. Flows support mechanism and asynchronous synchronization of the flows, one-way and request - response exchange - the model and other architectural plans.

**Asynch:**

\* The Async scope is a branch processing block that executes simultaneously with the main flow. The main flow continues to execute while it initiates and processes the Async scope. The flow does not have to pause until the last message processor embedded in the asynchronous flow has completed its task.

\* Async can be useful for executing time-consuming operations that do not require you to send a response back to the initiating flow (such as printing a file or connecting to a mail server).

\* To facilitate simultaneous branch processing, the Async scope sends one copy of the message it has received to the first embedded message processor in its own processing block. At the same time, it sends another copy of the message to the next message processor in the main flow.

\* Because the Async scope is executed in a "fire and forgot" manner, the result of the processing within the scope is not available in the main flow.

**File Connectors**

\* FileConnector acts as an outbound endpoint if you place a building block at the

middle or end of the Mule flow (basically in the Message Processor Region), passing

the file to the connected file system.

\* Place the FileConnector in the Message Processor of the Mule Flow to act as an

Outbound Endpoint

**Features:**

The File connector handles files and folders on a locally mounted file system. Its main features include:

\* The ability to read files or fully list directory contents on demand.

\* Support for locking files.

\* File matching functionality.

**File Configuration:**

\* The File connector does not always require configuration. However, it is good practice to define one. The most important configuration parameter is the Working Directory, which is the path to a directory that is treated as the root of every relative path that you specify with this connector.

\* If you do not provide a working directory, the connector configuration defaults to the value of the user.home system property.

**Read and Write to a file**

**Flow be like**:

On new or upload file -> Logger -> Transform message (Read a file)

On new or upload file -> Logger -> Transform message -> File write (Write to file)

**On new or upload file:**

\*It is an event source endpoint that holds input file which we want to give.

\* Sometimes it contains information by external and sometimes, we can give information in mule flow itself, it completely depends on requirement.

**Logger:**

This component helps you to monitor and debug your Mule application by logging important information such as error messages, status notifications and payload.

**Transform message:**

\* The Transform Message component carries out transformations over the input

data it receives. You can explicitly write out a transformation in DataWeave

language.

\* It is used to transform one format of data into any format of data.

**Flow Explanation:**

1. Take On new or upload file to give some format of data as input.

2. Take Logger to print the data on console and monitor mule application.

3. Take transform message to transform one format of data into any format of

data.

4. To read file, simply take three components because we’re not performing any

operation except reading a file which external resource.

5. To write a file take file write as endpoint connector because we want to store the output in a file which is file write here.