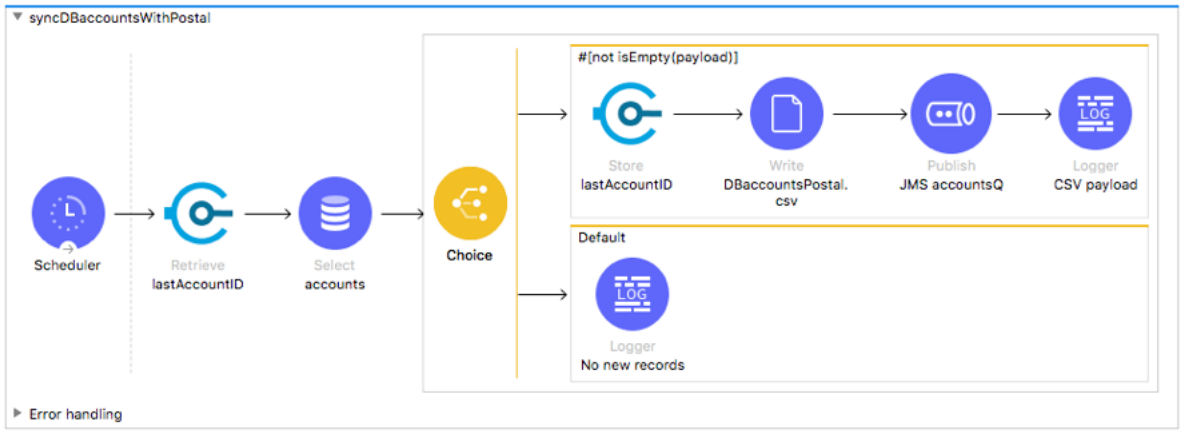
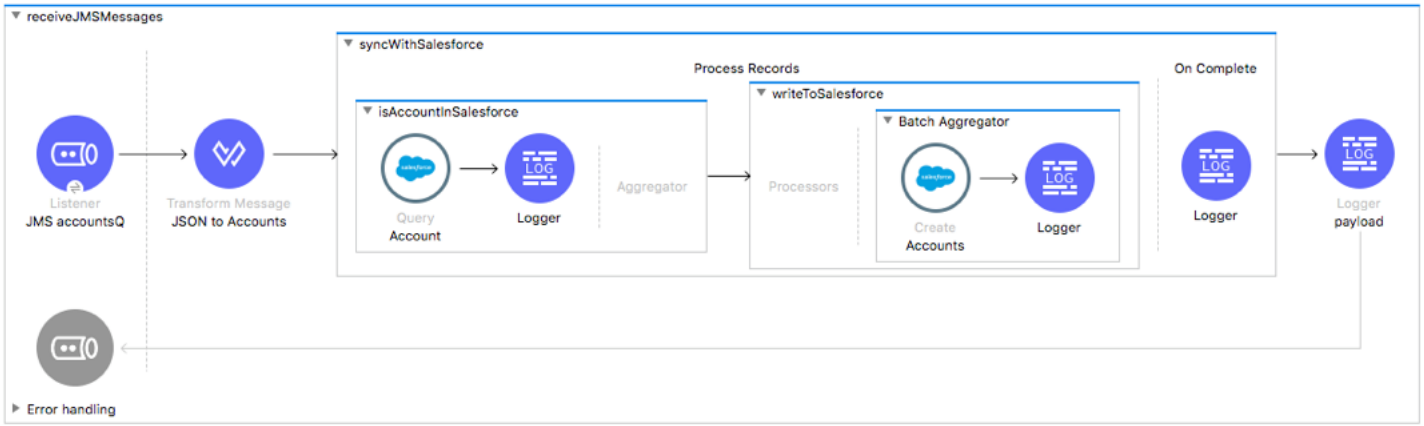
hello


MuleSoft
PART 3: Building Applications to Synchronize Data

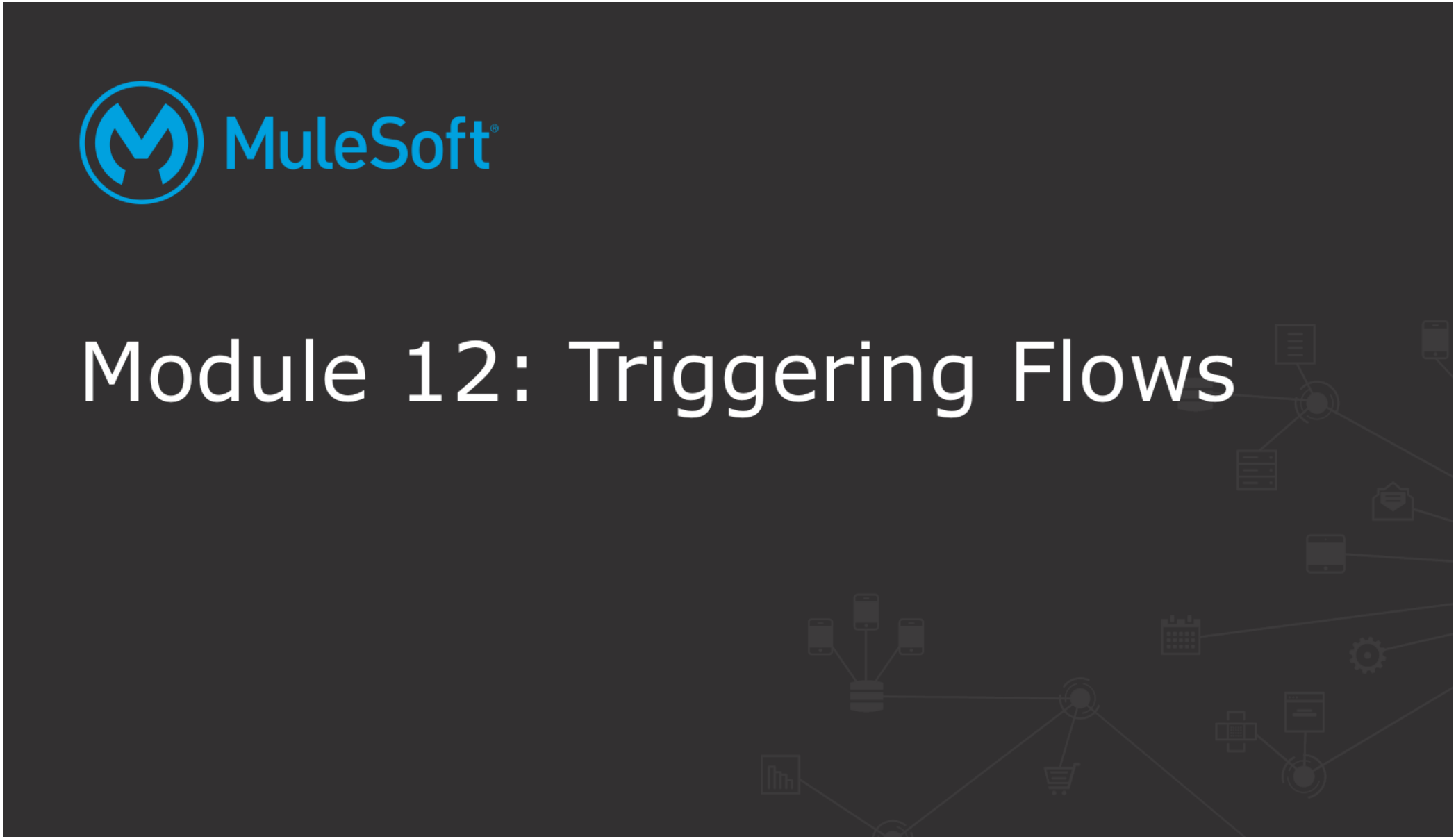
Goal





At the end of this part, you should be able to

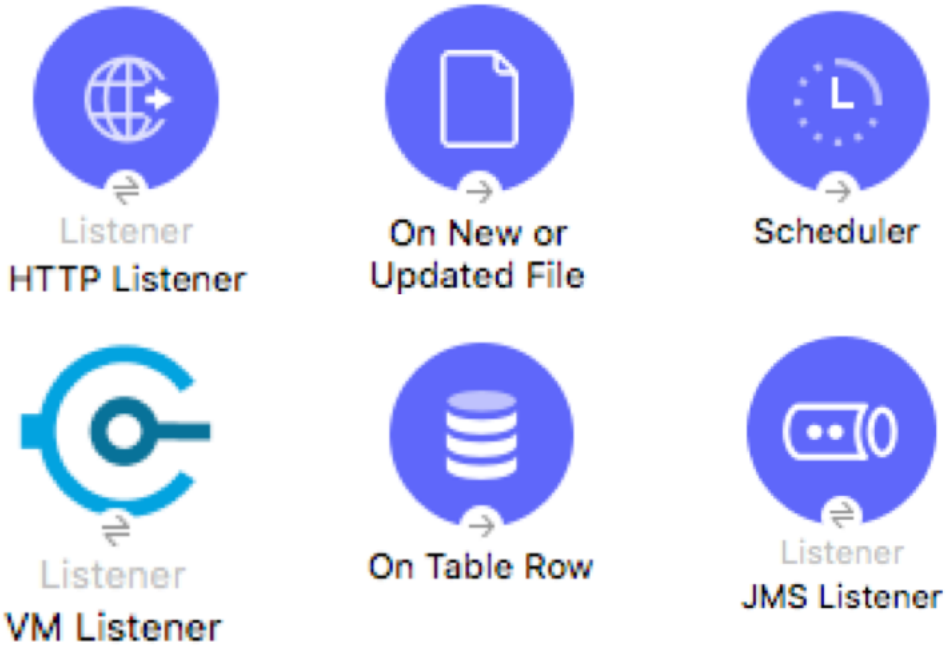
- Trigger flows when files or database records are added or updated
- Schedule flows
- Persist and share data across flow executions
- Publish and consume JMS messages
- Process items in a collection sequentially
- Process records asynchronously in batches


MuleSoft
Module 12: Triggering Flows

Goal

How have we initiated flows so far?

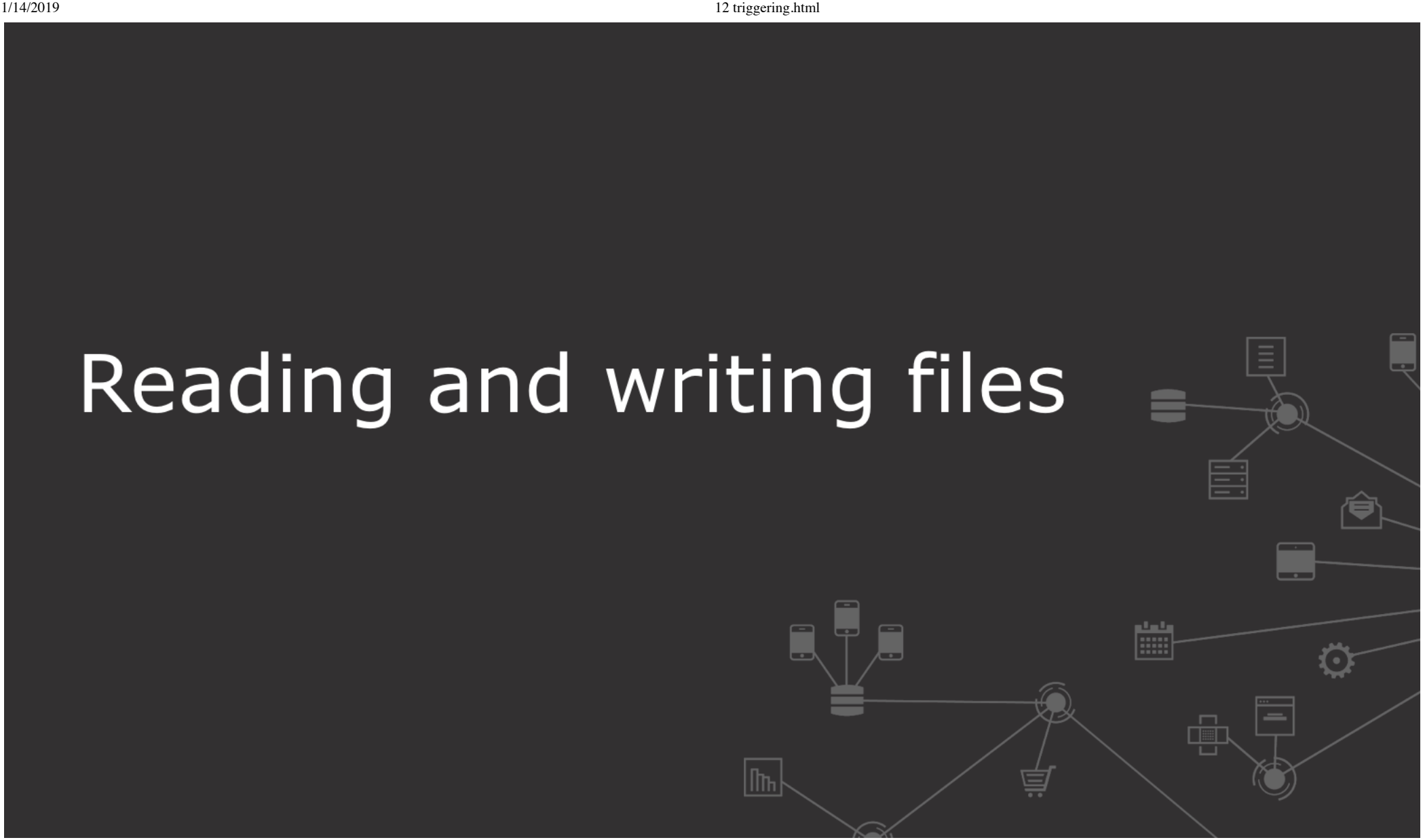
In this module, we will learn new ways



At the end of this module, you should be able to

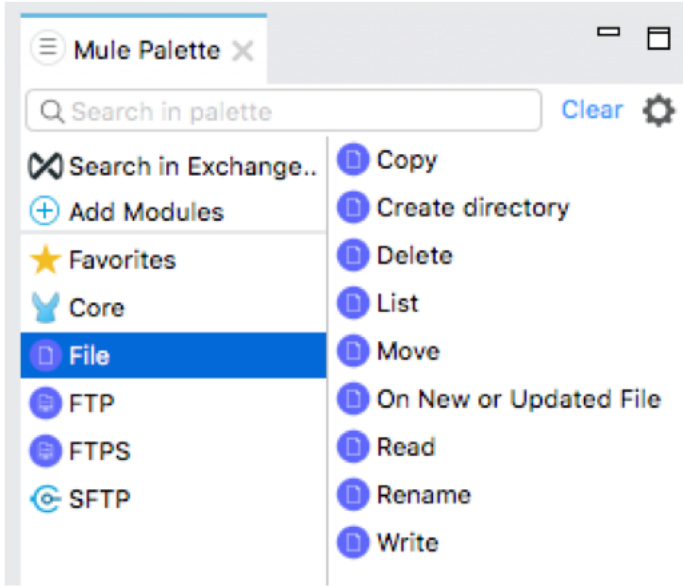
- Read and write files
- Trigger flows when files are added, created, or updated
- Trigger flows when new records are added to a database table
- Schedule flows to run at a certain time or frequency
- Persist and share data in flows using the Object Store
- Publish and consume JMS messages

Reading and writing files



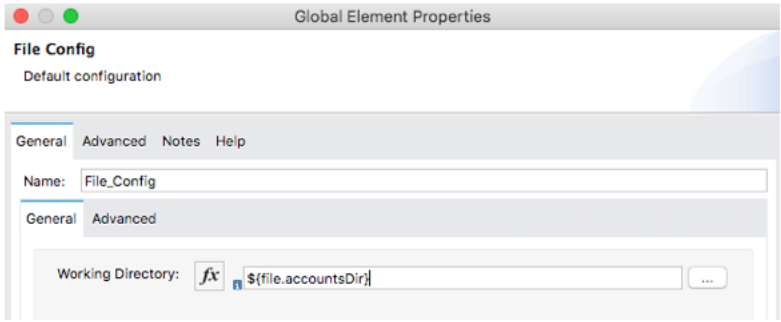
Reading and writing files

- There are 4 connectors for working with files and folders
  - File (for locally mounted file system)
  - FTP
  - FTPS
  - SFTP
- All have the same set of operations and they behave almost identically
- Support for
  - File matching functionality
  - Locking files
  - Overwriting, appending, and generating new files



Using the File connector

- Add the File module to the project
- Create a global element configuration
  - Not required but a best practice
  - Set the **working directory** that will be the root for every path used with the connector



- Use one of the connector operations and specify its properties

- On **CloudHub**, the connector can only be used with the /tmp folder
- On **Customer-hosted** Mule runtimes, the account running Mule must have read and/or write permissions on the specified directories
- *Be careful not to permanently delete or overwrite files*
  - Move or rename them after processing

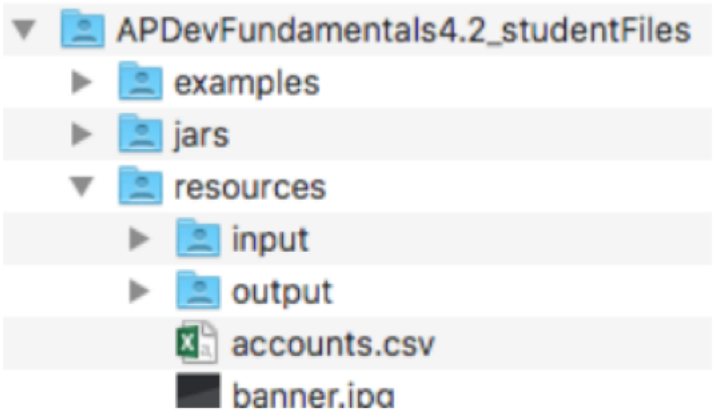
Trigger a flow when a new file is created or updated

- Use the **On New or Updated File** listener
  - Polls a directory for files that have been created or updated
  - One message is generated for each file that is found

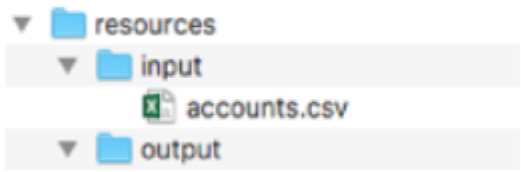
- Multiple ways to ensure a file is new
  - Delete each file after it has been processed so all files in the next poll will be new
  - Move each file to a different directory after it has been processed
  - Rename a file after it has been processed and filter the files to be processed
  - Save and compare the file creation or modification times

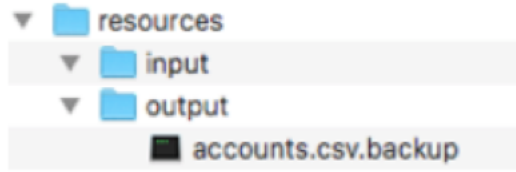
Walkthrough 12-1: Trigger a flow when a new file is added to a directory

- Add and configure a File listener to watch an input directory
- Restrict the type of file read
- Rename and move the processed files




```
111 Boulevard Hausmann,Paris,France,,Dog Park Industries,75008
400 South St,San Francisco,USA,CA,Iguana Park Industries,91156
777 North St,San Francisco,USA,CA,Cat Park Industries,91156
```





Synchronizing data with watermarks

Synchronizing data from one system to another

- The general process
  - The first time, you need to sync all the data
  - After that, you only need to sync the new data
- How do you determine what is new and needs to be synced?
  - On the first sync, store the latest timestamp for any item in the data set
  - On later syncs, retrieve that timestamp and compare the timestamp of each item and see if it is later
- The timestamp is often a
  - Creation timestamp
  - Modification timestamp
  - Record ID

Introducing watermarks

- The timestamp that is stored each sync and then retrieved and compared against in the next sync is called a **watermark**

- Where did the name come from?
  - After a flood, one might record how high the water got by marking the level on a wall
  - Similarly, for data, we want to look at the last value - how "high" it was in the last sync

Types of watermarking in Mule

- **Automatic**
  - The saving, retrieving, and comparing is automatically handled for you
  - Available for several connector listeners
    - On New or Updated File
    - On Table Row
  - Restricted in how you can specify what items/records are retrieved

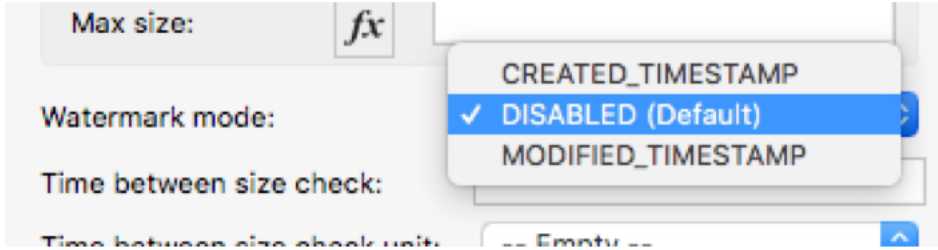
- **Manual**
  - You handle saving, retrieving, and comparing the watermark
  - More flexible in that you specify exactly what records you want retrieved

Using listeners with automatic watermarking

Using automatic watermarking with files

- There is a watermarking option for the **On New and Updated File** operation for the family of file connectors

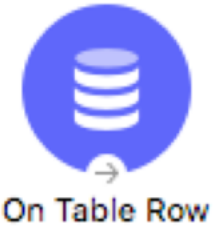
- There are two watermarking modes
  - CREATED_TIMESTAMP
  - MODIFIED_TIMESTAMP



- This can be used for one of the ways introduced last section to ensure a file is new
  - Other options: Delete, move, filter
  - **Save and compare the file creation or modification times**

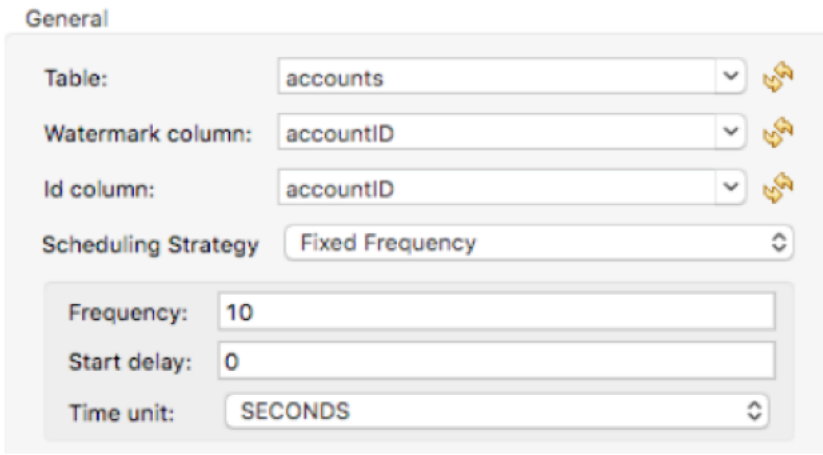
Triggering a flow for each row in a database table

- The Database connector has an **On Table Row** operation that is triggered for every row in a table



- The operation can handle
  - Generating the query
  - Watermarking
  - Idempotency across concurrent requests

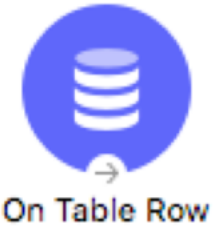
- You can specify one, both, or neither of
  - Watermark column
  - Idempotency column



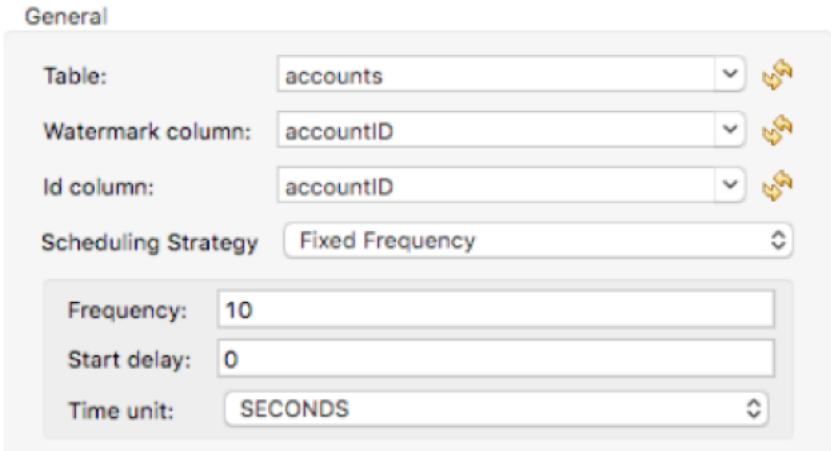
Using automatic watermarking with database tables

- When a watermark column is specified, this query is automatically generated and used

```
SELECT * FROM table
WHERE TIMESTAMP > :watermark
```



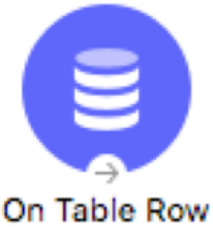
- On each poll, the component will go through all the retrieved rows and store the maximum value obtained



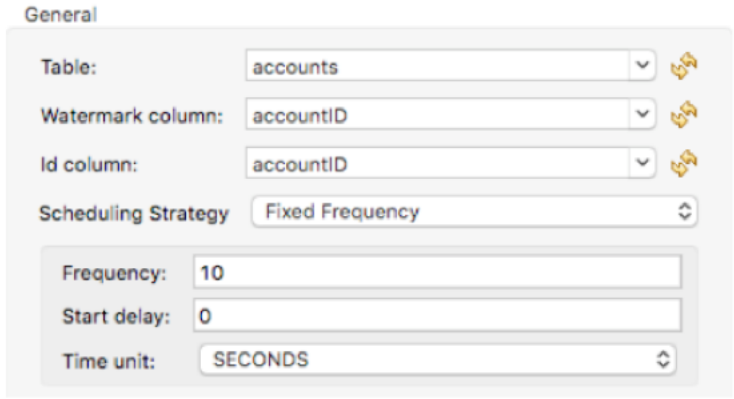
Handling idempotency across concurrent requests

MuleSoft

- **A new poll can be executed before the watermark is updated if**
  - The poll interval is small
  - The amount of rows is big
  - Processing one single row takes too much time



- **To avoid a record being processed more than once**
  - Specify an ID column
    - A unique identifier for the row
  - The listener will make sure the row is not processed again if
    - It has already been retrieved and
    - Processing of it hasn't finished yet



Walkthrough 12-2: Trigger a flow when a new record is added to a database & use automatic watermarking

- Add and configure a Database listener to check a table on a set frequency for new records
- Use the listener's automatic watermarking to track the ID of the latest record retrieved and trigger the flow whenever a new record is added
- Output new records to a CSV file
- Use a form to add a new account to the table and see the CSV updated

MUA Accounts

accountID	name	street	city
13388	Maxine Mule	415 Mission St	San Francisco
13387	Maximilian Mule	123 Main St	San Francisco

DBaccounts.csv

```
13384,United States,Peabody,15 Margin St,Joseph Silva,MA,01960
13385,Japan,Osaka,1753 Yashiki Way,Teruko Handa,Osaka,530-000
13386,United States,Gilmanton Iron Works,5 Barr Rd,Charles J Umanita,NH,03837
13387,United States,San Francisco,123 Main St,Maximilian Mule,CA,94111
13388,United States,San Francisco,415 Mission St,Maxine Mule,CA,94105
```

Using manual watermarking and scheduling flows

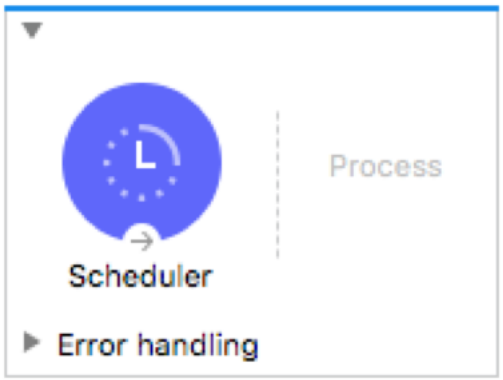
Handling watermarking manually

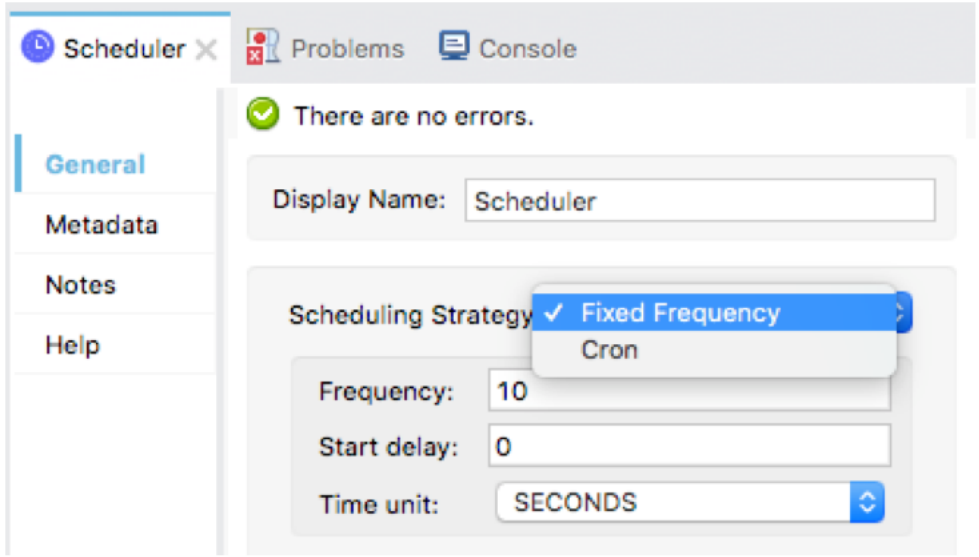
- The general process
  - Schedule when a flow should be executed
  - Give the watermark a default value
  - On the first sync
    - Determine a new watermark value
    - Store the watermark value so it available in the future to other flow executions
  - On later syncs
    - Retrieve the watermark from storage
    - Check if each item in the data set should be retrieved based on the watermark value

Triggering flows at a certain time or frequency

- Some connector operations use a scheduling strategy to trigger a flow
  - Like On New or Updated File and On Table Row

- To trigger **any** flow at **any** time, use the **Scheduler** component





Two types of scheduling strategies

- **Fixed frequency**
  - The default is to poll every 1000 milliseconds

- **Cron**
  - A standard for describing time and date information
  - Can specify either
    - An event to occur just once at a certain time
    - A recurring event on some frequency

0 15 10 ? * *	Poll at 10:15am every day
0 15 10 * * ? 2018	Poll at 10:15pm every day in 2018
1 1 1 1,6 *	Poll the first day of January and June every year in the first second of the first minute of the first hour

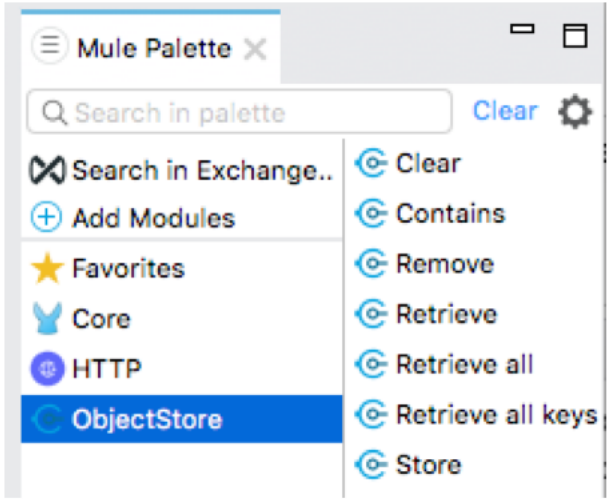
Persisting data across executions of flows

- Use the **Object Store** component to store simple key-value pairs
  - The component was designed to store
    - Synchronization information like watermarks
    - Temporal information like access tokens
    - User information
  - The values are accessible as event variables

- Each Mule application has an Object Store that is
  - Available without any setup or configuration
  - Persistent
    - Saved to file for embedded Mule and standalone Mule runtime
    - Saved to data storage for CloudHub
    - Saved to shared distributed memory for clustered Mule runtimes

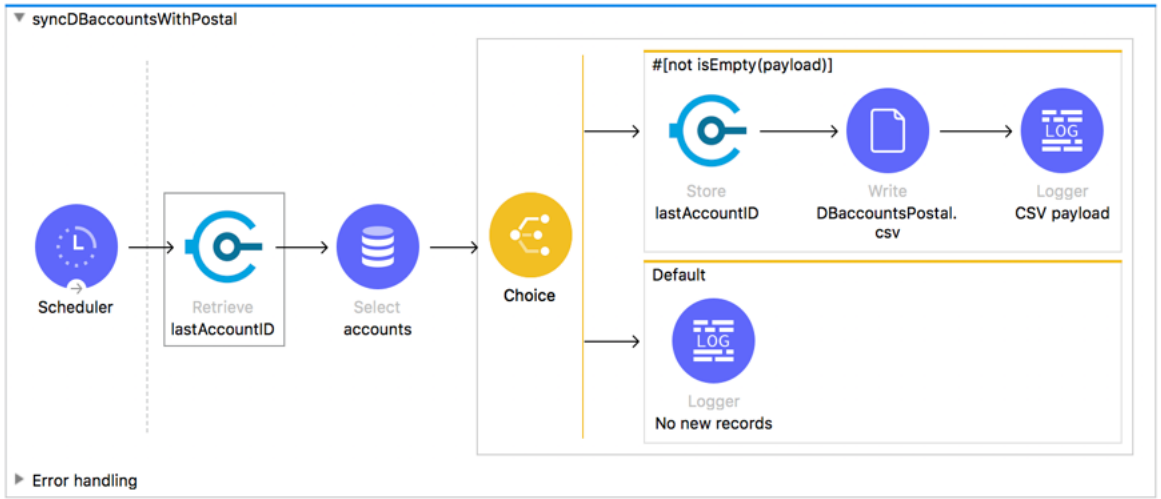
Using the Object Store connector for watermarking

- Add the ObjectStore module to the project
- Use the **Retrieve** operation to retrieve a watermark value and to assign a default value for the first poll
- Use the watermark value in a processor to retrieve the desired items
  - Like in a database query for records in a table
- Use the **Store** operation to determine and store a watermark value



Walkthrough 12-3: Schedule a flow and use manual watermarking

- Use the Scheduler component to create a new flow that executes at a specific frequency
- Retrieve accounts with a specific postal code from the accounts table
- Use the Object Store component to store the ID of the latest record and then use it to only retrieve new records



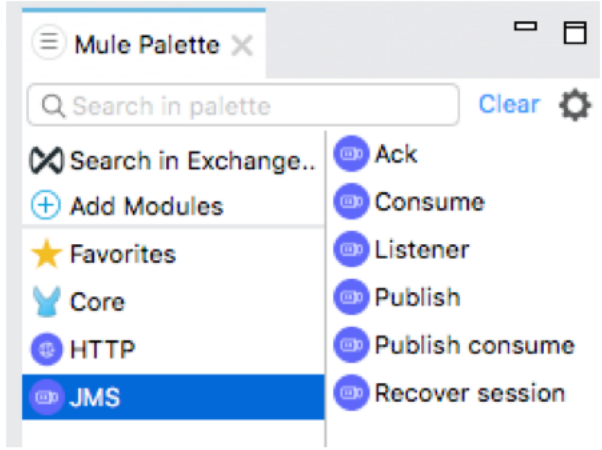
Publish and consume JMS messages

Java Messaging Service (JMS)

- Is a widely-used API for enabling applications to communicate through the exchange of messages
- Simplifies application development by providing a standard interface for creating, sending, and receiving messages
- Supports two messaging models
  - **Queues**: PTP (point-to-point or 1:1)
    - A sender sends messages to a queue & a single receiver pulls the message off the queue
    - The receiver does not need to be listening to the queue at the time the message is sent
  - **Topics**: Pub-Sub (publish/subscribe or 1:many)
    - A publisher sends a message to a topic & all active subscribers receive the message
    - Subscribers not actively listening will miss the published message (unless messages are made durable)

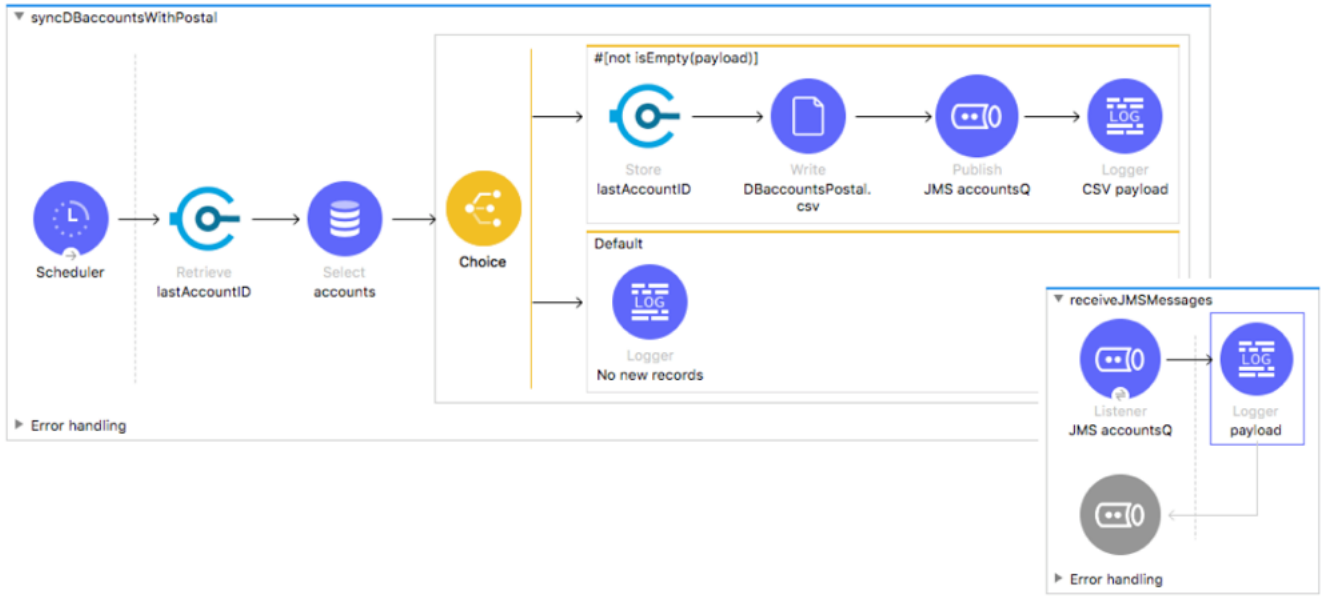
Using the JMS connector

- Add the JMS module to the project
- Configure a global element configuration
  - By default, it is set up with a finely tuned set of values for both for publishing and consuming messages
  - Typically, you just need to configure which connection should be used
- Use operations to publish and/or consume messages to destinations



Walkthrough 12-4: Publish and listen for JMS messages

- Add and configure a JMS connector for ActiveMQ (that uses an in-memory broker)
- Send messages to a JMS queue
- Listen for and process messages from a JMS queue



Summary

Summary

- Use **watermarks** to synchronize data across data stores
  - Use either **manual** or the **automatic** watermarking available for some connectors
- Use the family of **File**, **FTP**, **FTPS**, and **SFTP** connectors to work with files and folders
- Use the **On New or Updated File** listener to trigger flows when files are added, created, or updated
  - Use the connector's **automatic watermarking** to determine if a file is new based on a creation or modification timestamp
- Use the **On Table Row** listener when new records are added to a database table
  - Use the connector's **automatic watermarking** to determine if the record is new

Summary

- Use the **Scheduler** component to schedule flows to run at a certain time or frequency
  - Use a watermark to keep a persistent variable between scheduling events
- Use the **Object Store** connector to persist and share a watermark (or other data) across flow executions
- Use the **JMS** connector to publish and consume messages
  - Connect to any JMS messaging service that implements the JMS spec