



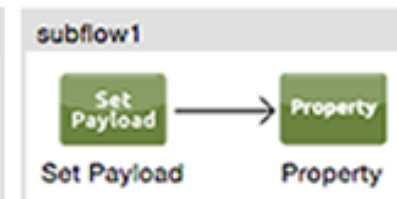
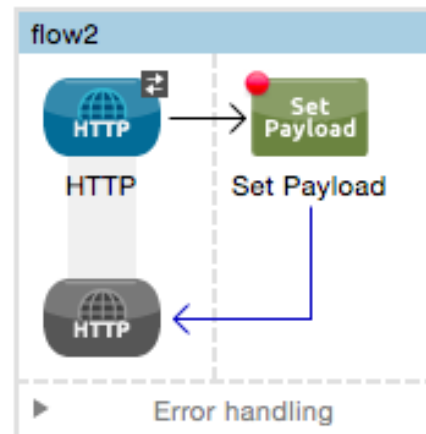
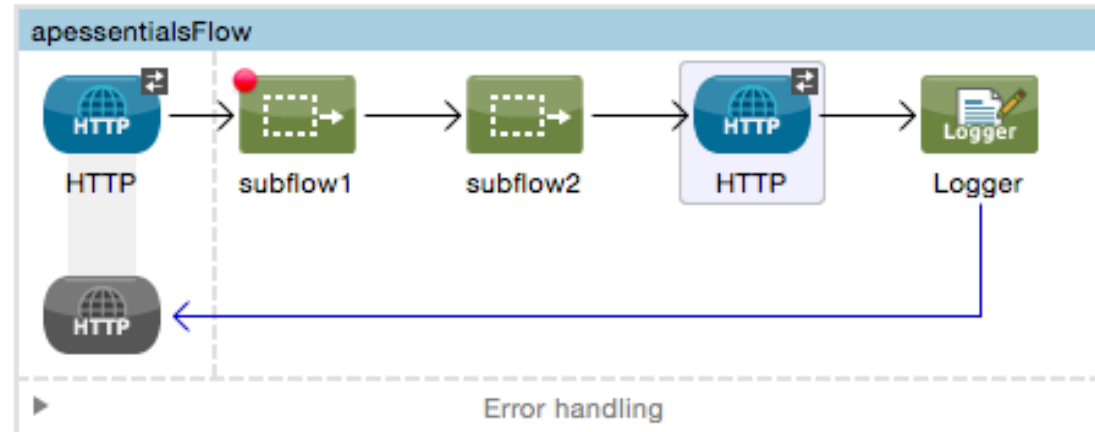
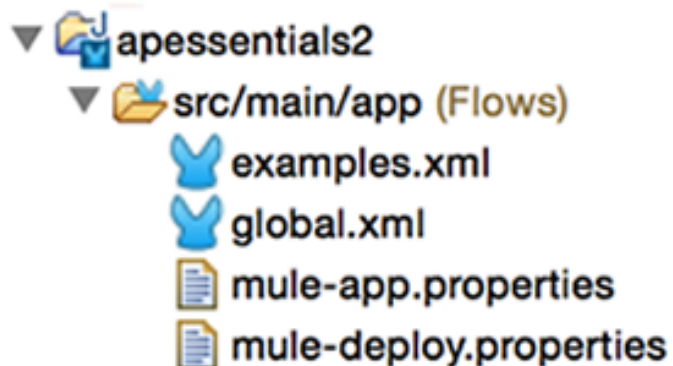
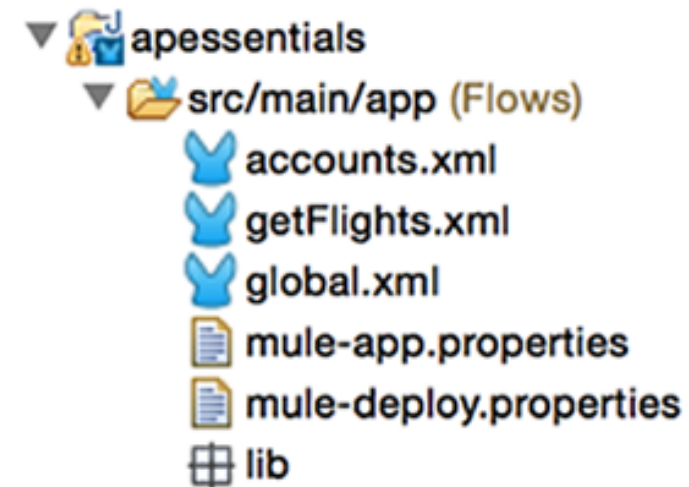
Module 6: Refactoring Mule Applications



Objectives

- In this module, you will learn:
 - To separate applications into multiple configuration files
 - To encapsulate global elements in a separate config file
 - To create and run multiple applications
 - To create and reference flows and subflows
 - About variable persistence through subflows and flows and across transport barriers

Goal



Organizing projects and applications

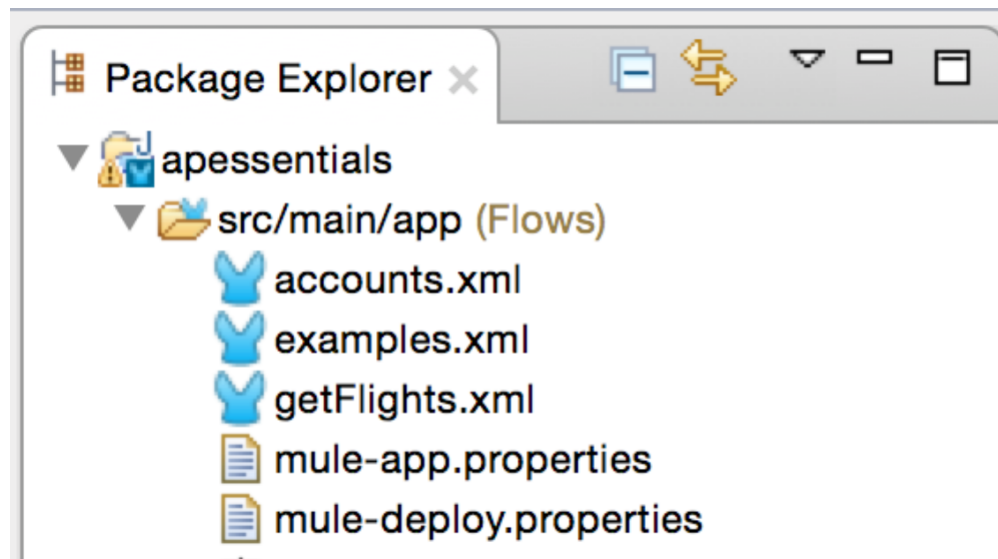


Separating applications into multiple configuration files

- Monolithic files are difficult to work with
- Separating an application into multiple configuration files makes code
 - Easier to read
 - Easier to work with
 - Easier to test
 - More maintainable

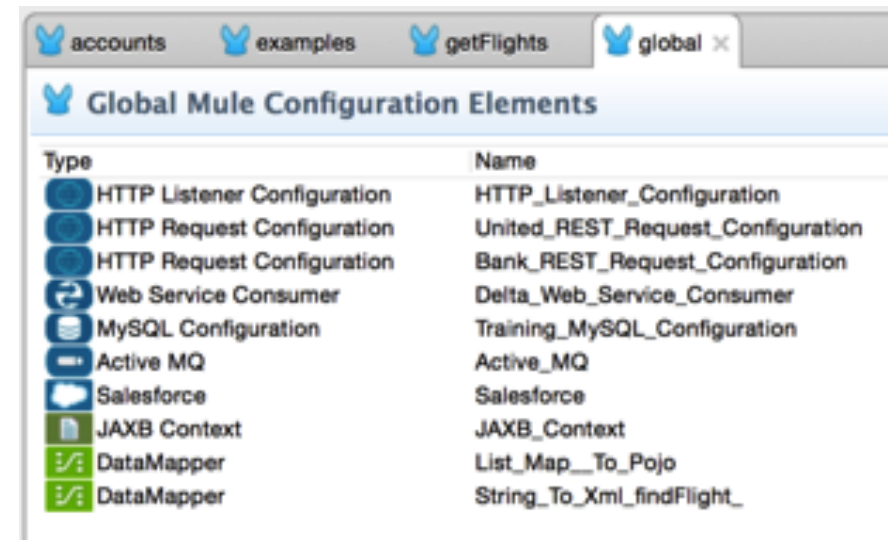
Walkthrough 6-1: Separate applications into multiple configuration files

- Separate the account flows into accounts.xml
- Move the rest of the example flows into examples.xml
- Rename apessentials.xml to getFlights.xml



Encapsulating global elements in a configuration file

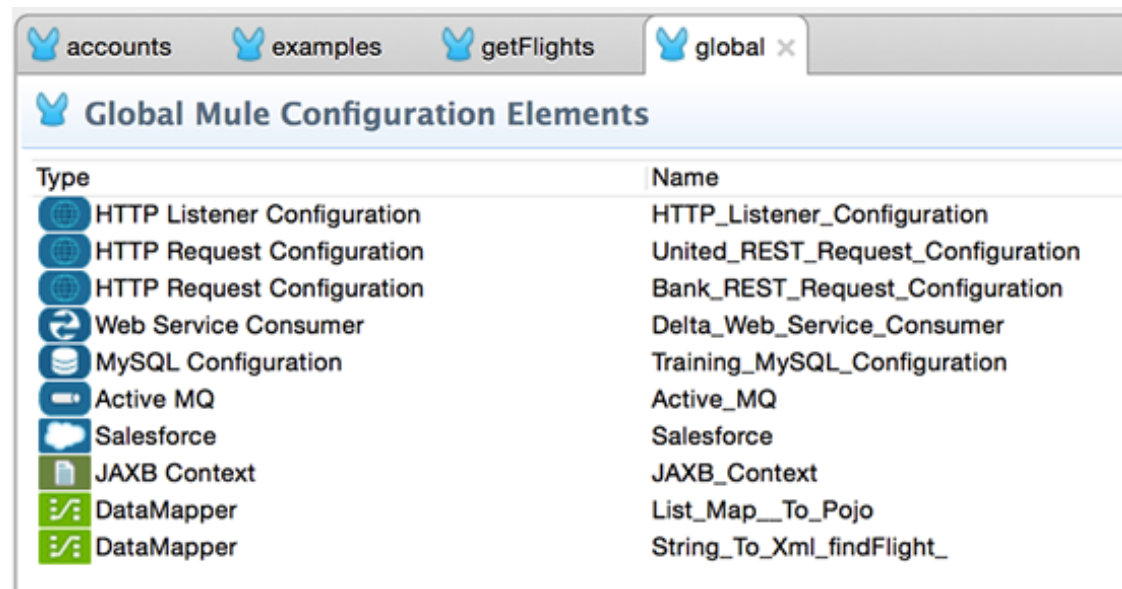
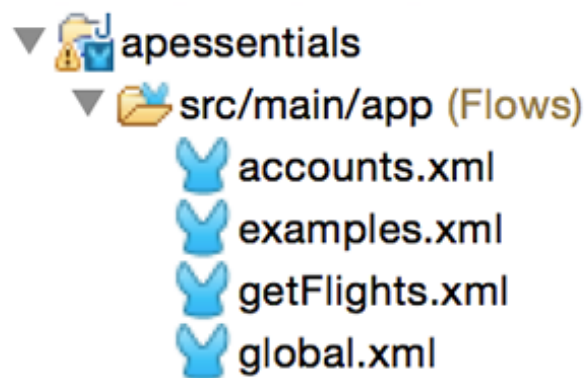
- It can be confusing if you reference global elements in one file that are defined in various, unrelated files
- It also makes it hard to find them
- A good solution is to put most global elements in one configuration file
 - All the rest of the files reference them
 - If a global element is specific to and only used in one file, it makes sense to keep it in that file
 - DataMapper configurations



Type	Name
HTTP Listener Configuration	HTTP_Listener_Configuration
HTTP Request Configuration	United_REST_Request_Configuration
HTTP Request Configuration	Bank_REST_Request_Configuration
Web Service Consumer	Delta_Web_Service_Consumer
MySQL Configuration	Training_MySQL_Configuration
Active MQ	Active_MQ
Salesforce	Salesforce
JAXB Context	JAXB_Context
DataMapper	List_Map_To_Pojo
DataMapper	String_To_Xml_findFlight

Walkthrough 6-2: Encapsulate global elements in a separate configuration file

- Create a configuration file for just global elements
- Move all the existing global elements to this file

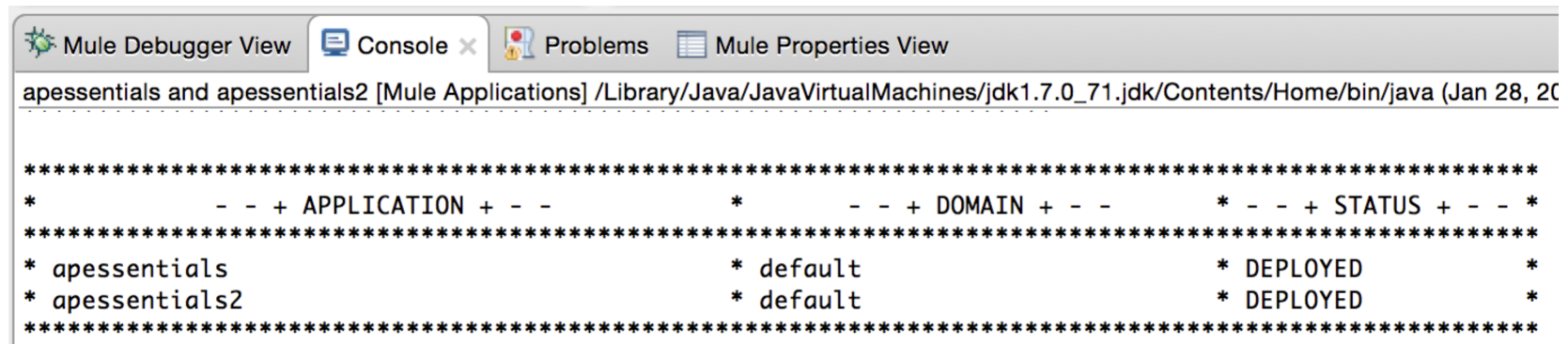


Creating multiple applications

- Separate functionality into multiple applications to
 - Allow managing and monitoring of them as separate entities
 - Use different, incompatible JAR files
- Run more than one application at a time in Studio by creating a run configuration

Walkthrough 6-3: Create and run multiple applications

- Create a new project and application called apessentials2
- Move the examples config file into the apessentials2 application
- Create new global connector configurations in the new project
- Create a new run configuration to run multiple applications simultaneously



```
Mule Debugger View Console Problems Mule Properties View
apessentials and apessentials2 [Mule Applications] /Library/Java/JavaVirtualMachines/jdk1.7.0_71.jdk/Contents/Home/bin/java (Jan 28, 2015)

*****
*      - - + APPLICATION + - -      *      - - + DOMAIN + - -      *      - - + STATUS + - - *
*****
* apessentials                      * default                      * DEPLOYED                      *
* apessentials2                     * default                      * DEPLOYED                      *
*****
```

Sharing resources between applications

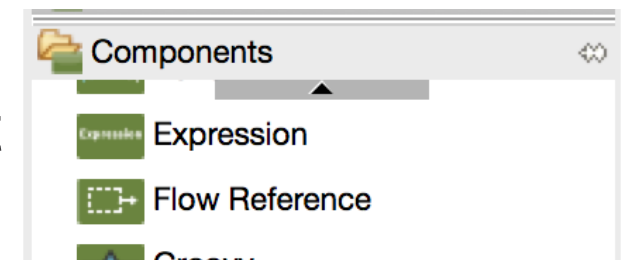
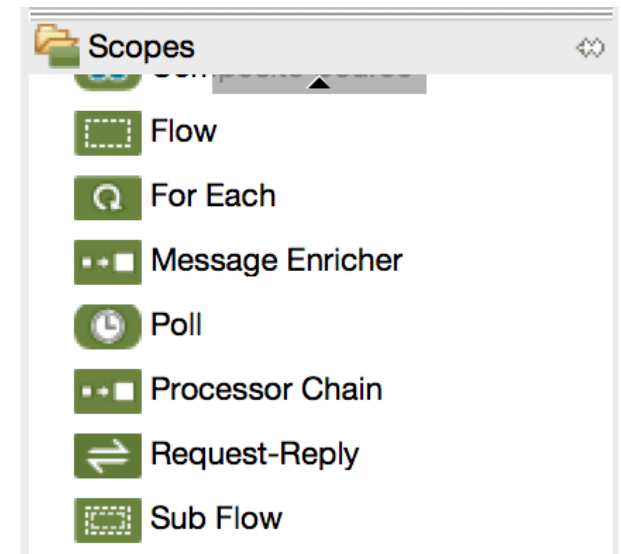
- Use domains and shared resources
 - Create a Mule Domain Project and associate Mule applications with a domain
 - Define a set of resources (and the libraries required by those resources) for a domain to share between the applications that belong to the domain
 - Only available on-prem, not on CloudHub

Encapsulating processors into separate flows



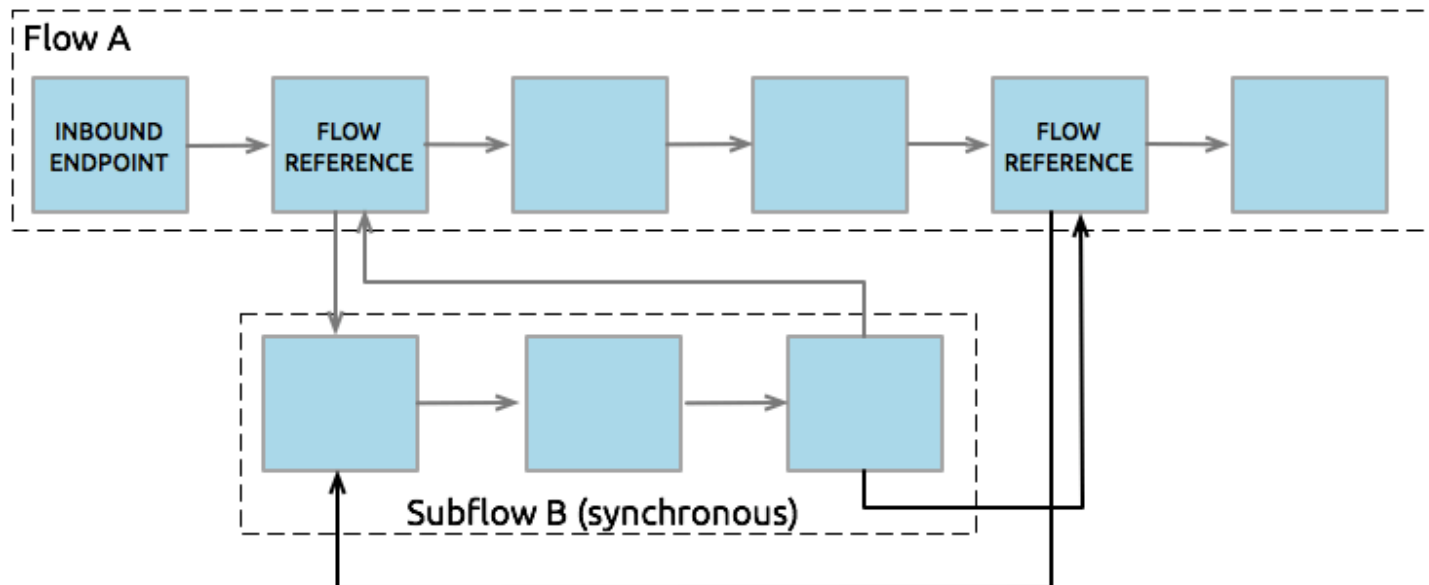
Flows and subflows

- Flows can be broken into multiple flows and subflows
 - Makes the graphical view more intuitive and the XML code easier to read
 - Promotes code reuse
- All flows are identified by name and can be called via Flow Reference components in other flows
- Flow variables persist through all flows unless the message crosses a transport boundary



Subflows

- Subflows are executed exactly as if the processors were still in the calling flow
 - Always run synchronously
 - Inherit the processing and exception strategies of the flow that triggered its execution

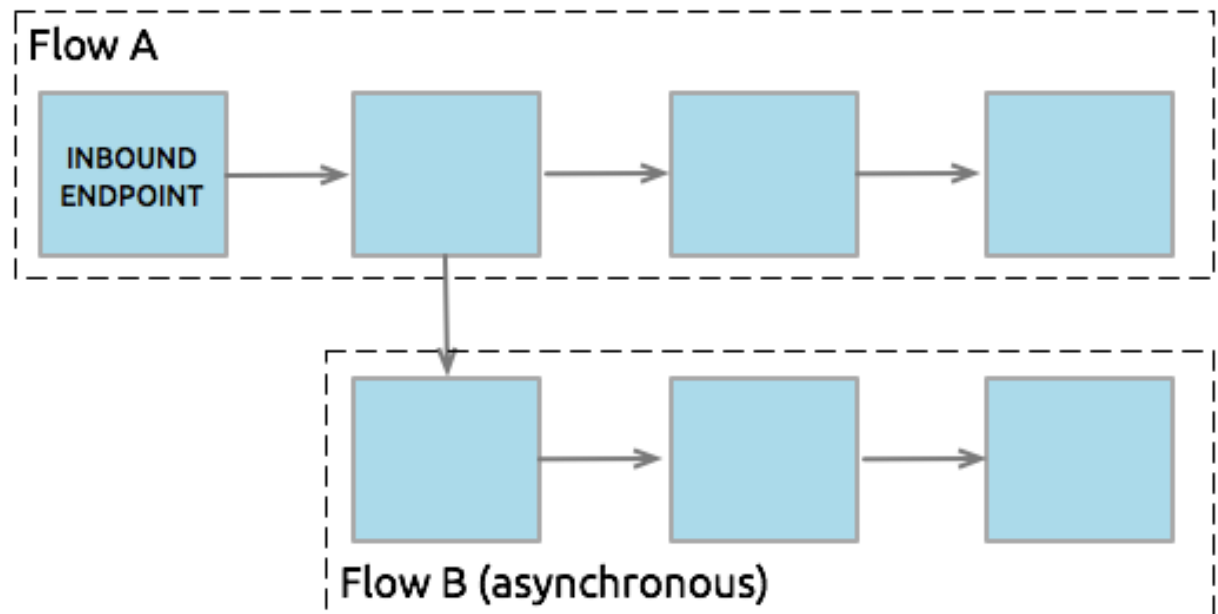
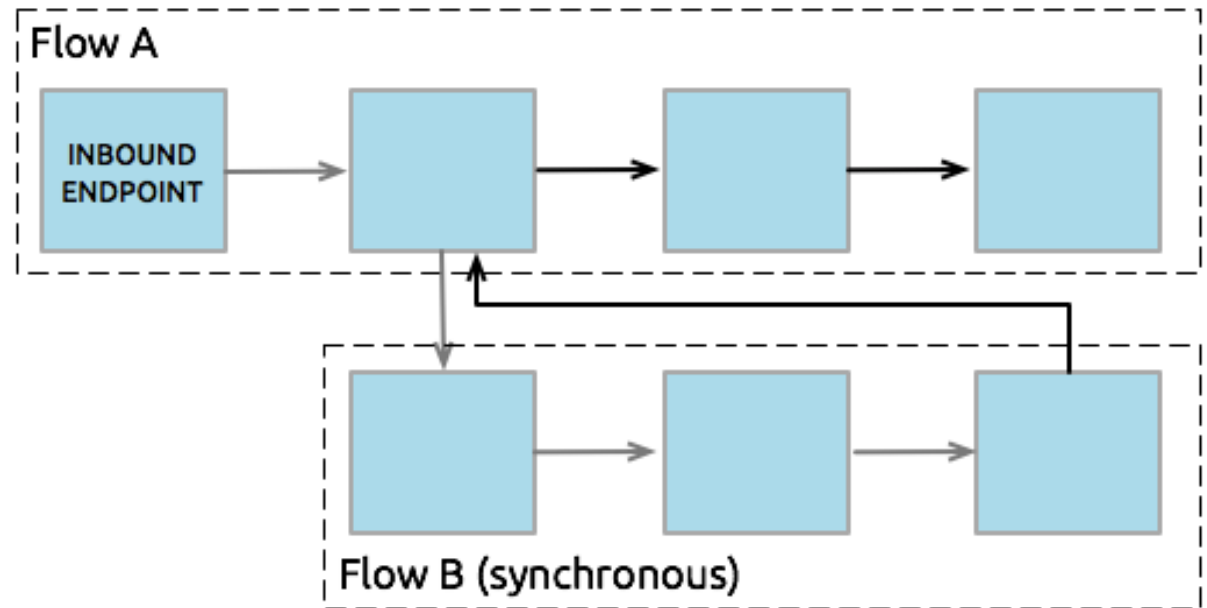


Flows

- Flows, on the other hand, have much more flexibility in how they are used
- Can have their own processing and exception strategies
- Can be synchronous or asynchronous
- Flows without message sources are sometimes called private flows

Synchronous and asynchronous flows

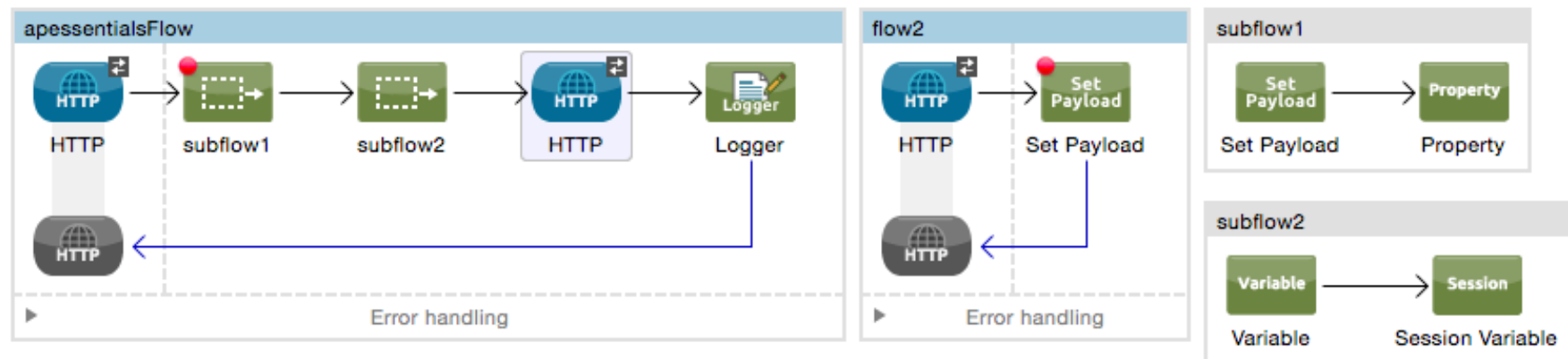
- Synchronous
 - Triggering flow pauses, then resumes only after the synchronous flow completes its processing and hands the message back to the triggering flow
- Asynchronous
 - A copy of the message is passed both to the asynchronous flow and simultaneously to the next message processor in its own flow
 - The two flows execute simultaneously and independently, each finishing on its own



Walkthrough 6-4: Create and reference flows and subflows

- Extract processors into separate subflows and flows
- Use the Flow Reference component to reference other flows
- Explore variable persistence through flows, subflows, and across transport barriers

Note: Session variables are persisted across some but not all transport barriers



Summary



Summary

- In this module, you learned to refactor Mule applications
- Separate application functionality into multiple configuration files for easier development and maintenance
- Encapsulate global elements that will be used in multiple config files into their own separate config file
- Separate functionality into multiple applications to allow managing and monitoring of them as separate entities
- Run more than one application at a time in Studio by creating a run configuration
- Share resources between applications by creating a shared domain

Summary

- Separate a flow into multiple flows for easier to read files and for reusability
- All flows are identified by name and can be called via Flow Reference components in other flows
- Flow variables persist through all flows unless the message crosses a transport boundary
- Subflows are executed exactly as if in the calling flow
 - Always run synchronously and inherit processing and exception strategies of the calling flow
- Flows can have their own processing and exception strategies and be synchronous or asynchronous