

**MuleSoft**

**Learning Program  
2016**

**People matter, results count.**



# Objectives of MuleSoft

- Purpose:
  - Understanding various integration pattern, and when to use the pattern.
- Product:
  - Understanding Integration Styles
  - Mule Soft implementation
- Process:
  - Learn Various Integration Styles
  - Connecters
  - Routers
  - Transformers
  - Filters

# Table of Contents

---

- Introduction
- Integration Pattern
- Integration Styles
- Terminologies of Integration
- Mule Soft Components
  - Connector
  - Router
  - Transformer
  - Filter
  - Exception Handling

# Introduction

- Runtime engine of Anypoint Platform.
- Lightweight Java-based enterprise service bus (ESB)
- Integration platform that allows developers to connect applications together quickly and easily, enabling them to exchange data.
- Easy integration of existing systems, regardless of the different technologies that the applications use, including JMS, Web Services, JDBC, HTTP, and more.
- The ESB can be deployed anywhere, can integrate and orchestrate events in real time or in batch, and has universal connectivity.

# Introduction

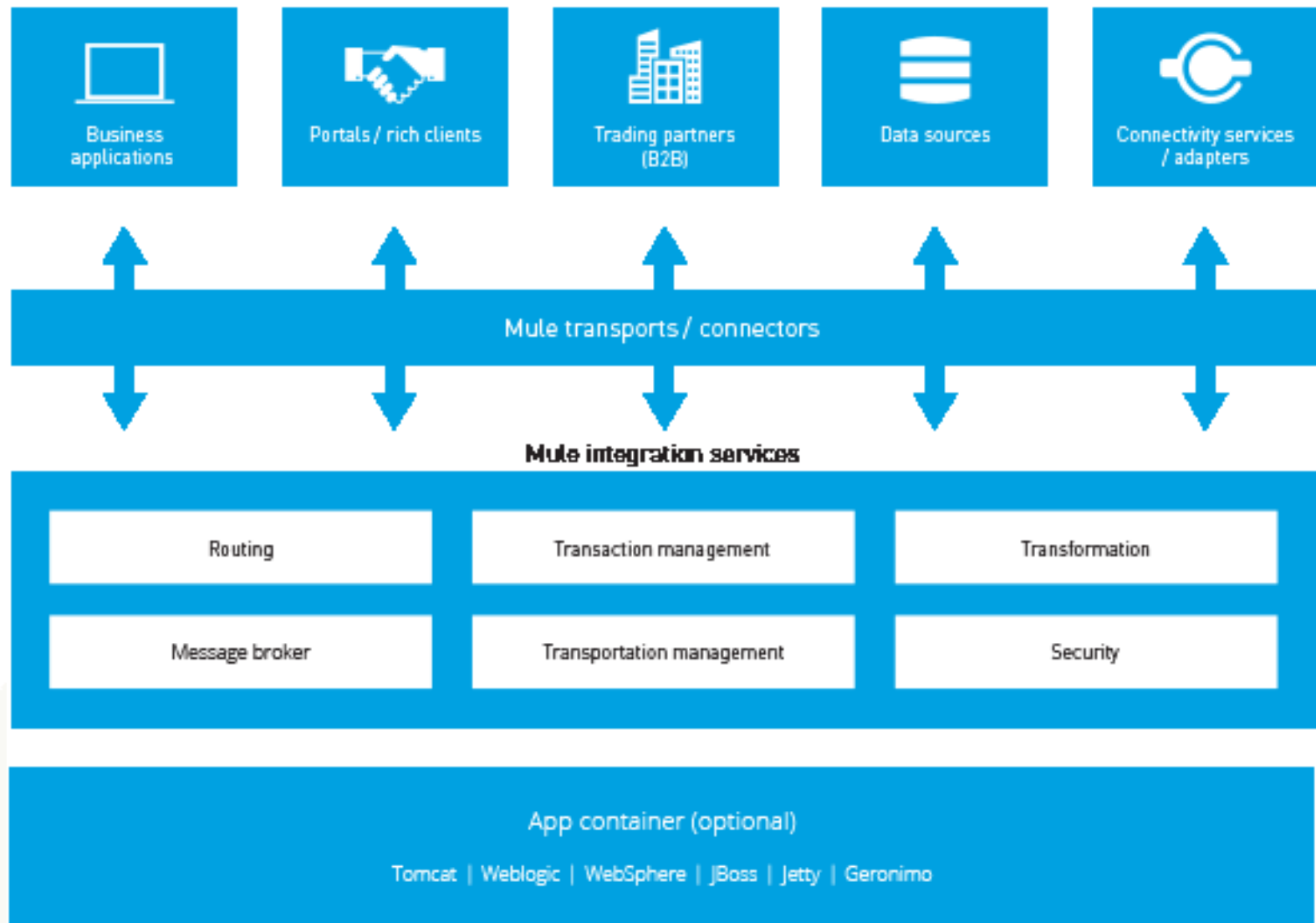
All integration solutions have to deal with a few fundamental challenges:

- Networks are unreliable
- Networks are slow
- Any two applications are different
- Change is inevitable

# Introduction

- Transit system for carrying data between applications within our enterprise or across the Internet.
- Mule has powerful capabilities that include:
  - Service creation and hosting
  - Service mediation
  - Message routing
  - Data transformation

# Mule ESB



# Integration Pattern

- Solving Integration Problems using Patterns
  - Information Portal
  - Data Replication
  - Shared Business Function
  - Service Oriented Architecture
  - Distributed Business Process
  - Business to Business
- Tightly Coupled System
- Loosely Coupled System



# Integration Styles

## ■ Integration Styles

- File Transfer
- Shared Database
- Remote Procedure Invocation
- Messaging

# Terminologies of Integration

- **Splitter**, a component that breaks a single message into multiple individual messages.
- **Aggregator**, the component that can combine multiple messages into a single message.
- A Content **Enricher** is a component that adds missing data items to an incoming message.
- The combination of a *Splitter*, a *Router* and an *Aggregator* is fairly common. We refer to it as a **Composed Message Processor**.
- **Process Manager** component that receives a New Order message (which includes the current shipping and billing address) and publishes two separate messages to the billing (or shipping) system
- A **Dynamic Recipient List** is the combination of two Message Routing patterns.
- **Message Store** can provide us with some important business metrics such as the average time to fulfill an order.

# Connector

- A **connector** is the object that sends and receive messages on behalf of an endpoint.
- Connectors are bundled as part of specific transports of providers.
- For example, the **FileConnector** can read and write file system files.
  - File
  - Database
  - VM
  - Anypoint MQ

# VM

- VM are in memory queues available in Mule JVM.
- Pros:
  - Can be used where message queuing setup is not unavailable in the infrastructure.
  - High performance compared to other message queue.
  - Asynchronies kind of transfer VM is best choice.
- Cons:
  - VM file persistency
  - Extended Memory Usage

# Router

- A **Router** is an object that do something with messages once they have been received by a connector, or prior to being sent out by the connector.
- Routers route messages to various destinations in a Mule flow. Some routers incorporate logic to analyze and possibly transform messages before routing takes place.
- Example
  - Scatter – Gather
  - Splitter
  - Choice

**Note:** In Anypoint routers are configured under **FlowControl**.

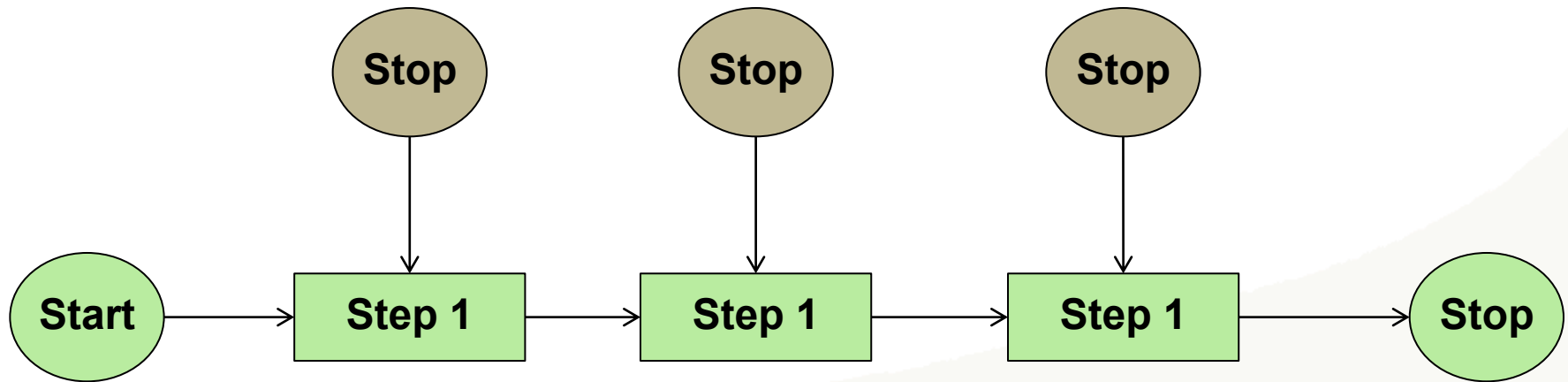
# Filter

- A filter optionally filters incoming or outgoing messages that are coming into or going out from a connector.
- For example, the File Provider comes with a **FilenameWildcardFilter** that restricts which files are read by the connector based on file name patterns. For example only files with the .xml extension can be routed.
- Filters are used in conjunction with Routers.

# Transformer

- A **transformer** optionally changes incoming or outgoing messages in some way
- This is usually done to make the message format useable by a downstream function
- Examples:
  - The **ByteArrayToString** transformer converts byte arrays into String objects.

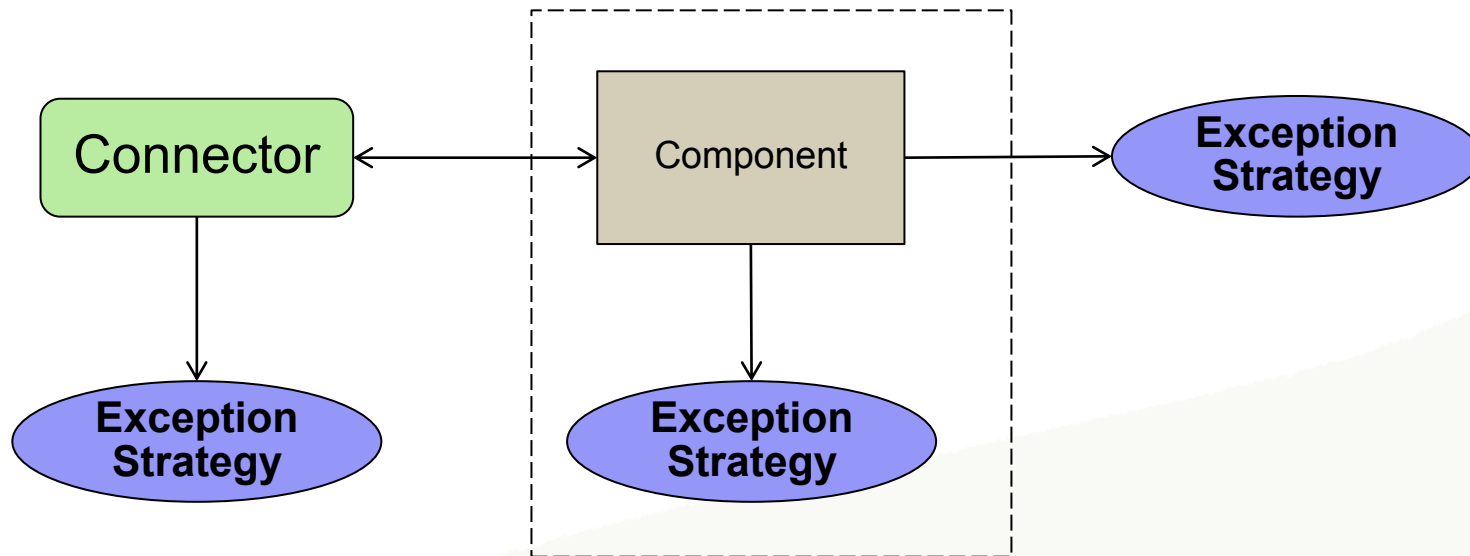
# Exception Handling



- By default, error free documents follow a central path known as the "happy path"
- Documents that have errors may be handled in different ways (rejected, warning etc.)



# Exception Handling



- Mule has a special way of handling non-happy path processing. This is called an "exception Strategy" but is it really just an exception path and there is very little strategy involved.
- There are three places you can associate an exception strategy
  - Connector
  - Component
  - Model(set for all components in a model)

# Exception Strategy

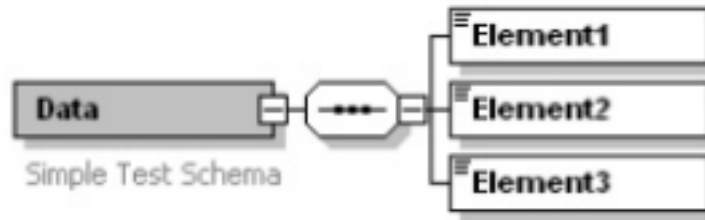
```
<exception-strategy  
  className="org.mule.impl.DefaultComponentExceptionStrategy">  
<endpoint address="file:///c:/mule-class/error" />  
</exception-strategy>
```

**we want all invalid documents to be moved into the error folder.**

# Exception Strategy

## Sample XML:

Given the following XML Schema file:



XML Schema validation will generate an error message when it gets to the fourth invalid data element:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Data>
  <Element1>Hello world</Element1>
  <Element2>String</Element2>
  <Element3>String</Element3>
  <BOIInvalidDataElement>
This is BOI invalid data element.
  </BOIInvalidDataElement>
</Data>
```

# Exception Strategy

```
<outbound-router>
<router className="org.mule.routing.outbound.FilterXmlMessageSplitter">
<endpoint address="file:///c:mule-class/out?outPattern=$[ORIGINALNAME]" />
  <properties>
    <property name="validateSchema" value="true"/>
    <property name="externalSchemaLoaction"
      value="file:///c:mule-class/labs/07-validate/my-schema.xsd"/>
  </properties>
</router>
</outbound-router>
```

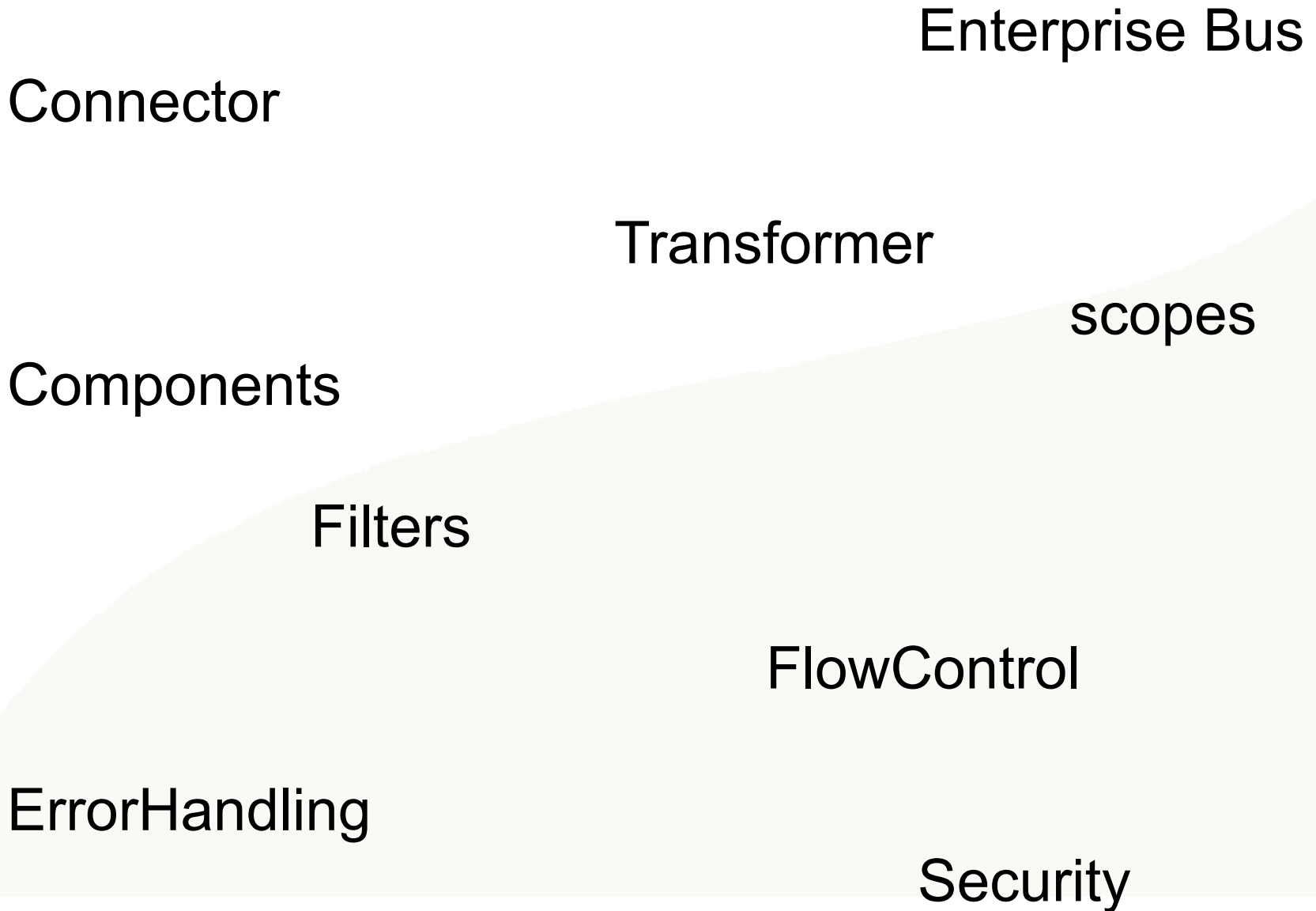
- To validate the XML Schema, just add two properties:
  - tell it to validate the document
  - tell it to what file to use and where to find it

# Exception Strategy

document : cvc-complex-type.2.4.d : Invalid content was found starting with element 'BOIInvalidDataElement'. No child element is expected at this point.

- This error message is generated on the Mule Console when an invalid data element is found. But what should we do with it? How do we redirect it to the appropriate user?

# Recap



Thank You For Your Time



People matter, results count.

