



MuleSoft

Anypoint Platform Architecture: Integration Solutions



Introductions



- Name
- Company & role
- Experience with integration architecture
- Experience with Anypoint Platform and MuleSoft products
- What do you plan on architecting on Anypoint Platform?
- What do you want to get out of class?

- Time
 - Class is for 5 days, typically from **9 to 5**
 - 1 hour lunch/mid-class break, typically from **12 to 1**
 - **Break(s)** each morning and afternoon
 - Other breaks as desired - just ask!
- We know you have two jobs to do this week!
 - If you have scheduled meetings, please let me know
 - We can try to schedule breaks around them

Introducing the course



- Anypoint Platform Architecture: **Integration Solutions** and MuleSoft **Certified Integration Architect** - Level 1
 - Drive and be responsible for an organization's Anypoint Platform implementation and the technical quality, governance (ensuring compliance), and operationalization of the integration solutions
 - Work with technical and non-technical stakeholders to translate functional and non-functional requirements into integration interfaces and implementations
- Anypoint Platform Architecture: **Application Networks** and MuleSoft **Certified Platform Architect** - Level 1
 - Define and be responsible for an organization's Anypoint Platform strategy
 - Direct the emergence of an effective application network out of individual integration solutions following API-led connectivity across an organization

The overall course goal

- Within the context of a particular use case, be able to
 - **Understand** and **model** features, options, and tradeoffs to
 - **Select** and document **meaningful** and **useful** design specifications to
 - **Balance**
 - Various project **requirements**
 - Sometimes **conflicting goals** of various technical stakeholders
 - **Other limitations and tradeoffs**

- **Anypoint Platform Architecture: Integration Solutions**

- **Solution and technical architects or lead/senior developers**

- With experience developing and deploying non-trivial Mule applications
 - Focused on designing enterprise integration solutions
 - Experienced in common integration approaches (like SOA) and integration technologies/platforms

- **Anypoint Platform Architecture: Application Networks**

- **Senior solution and enterprise architects**

- With basic knowledge and experience with the components of Anypoint Platform
 - Experienced in common integration approaches (like SOA) and integration technologies/platforms

Course prerequisites and training pathways

- Experience developing and deploying Mule applications as demonstrated by one of the following

- Passing the *MuleSoft Certified Developer - Level 1 (Mule 4)* exam
 - Completion of *Anypoint Platform Development: Fundamentals (Mule 4)*
 - Completion of *MuleSoft.U Development Fundamentals (Mule 4)*

- It is also helpful to have completed one or more of the following courses

- *Anypoint Platform Architecture: Application Networks*
 - *Anypoint Platform Operations: CloudHub*
 - *Anypoint Platform Operations: Customer-Hosted Runtimes*

Prerequisites: Development and architecture knowledge and experience



- Proficiency in any JVM-based programming language with ability to read procedural, object-oriented, and (ideally) functional code
- Familiarity with threads, thread pools, locks, server/client sockets, JDBC data sources, and connection pools on the JVM
- Proficiency with current software development tools like Git/GitHub, Maven, Jenkins, or similar

Prerequisites: Integration architecture experience



- Experience as an architect or lead/senior developer on at least one integration project using any technology stack
- A full understanding of the fundamental ingredients of enterprise integration including
 - Interface definitions and contracts
 - Data encoding using XML or JSON
 - REST APIs or SOAP web services
 - SQL or NoSQL database access
 - Message-passing using JMS, AMQP or similar
 - Network protocols like TCP/IP, HTTP and HTTPS
 - Single-resource transactions
- Familiarity with basic security concepts including certificates and encryption at rest and in transit

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



- Successfully carry out the various **job tasks** required of an integration solution architect to
 - Design integration solutions
 - Operationalize integration solutions
 - Deployment, logging, management, maintenance
 - Design and communicate non-functional requirements
 - Way to organize our thinking about cross cutting concerns that affect many/all applications
 - For example, security, scaling, reliability

Job tasks covered in the class



- Work with **technical** and **non-technical** stakeholders to translate **functional** and **non-functional requirements** into well documented **integration interfaces** and **detailed implementation designs**
- Guide implementation teams on the choice of **Mule components** and **patterns** to use in the **implementation of integration solutions designs**
- Design reusable **assets**, **components**, **standards**, **frameworks**, and **processes** to support and facilitate API and integration projects
- Apply standard development methods covering the **full development lifecycle** (project preparation, analysis, design, development, testing, deployment, and support) to **ensure solution quality**

- Design Mule applications for any of the available Anypoint Platform **runtime planes**
- Select the **deployment approach** and **configuration** of **Anypoint Platform** with any of the available **deployment options** (MuleSoft-hosted or customer-hosted control plane and runtime plane)
- Design and be responsible for the **technical quality, governance** (ensuring compliance), and **operationalization** of the integration solution
- Advise technical teams on **performance, scalability, reliability, monitoring** and **other operational concerns** of the integration solution on Anypoint Platform

Course outline - Part 1: Designing Integration Solutions

- Module 1: Architecting Integration Solutions
- Module 2: Identifying the Components and Capabilities of Anypoint Platform
- Module 3: Designing Integration Solutions using Mule Applications
- Module 4: Choosing Appropriate Mule 4 Processing Models
- Module 5: Choosing Mule Event Transformation and Routing Patterns
- Module 6: Designing Testing Strategies for Mule Applications

Course outline - Part 2: Designing Operationalization of Integration Solutions



- Module 7: Choosing and Developing a Deployment Strategy
- Module 8: Designing State Preservation and Management Options
- Module 9: Designing Effective and Sufficient Logging and Monitoring
- Module 10: Creating an Efficient and Automated Software Development Lifecycle

Course outline - Part 3: Designing to Meet Non-Functional Requirements



- Module 11: Designing for Transactional Requirements
- Module 12: Clarifying and Designing for Reliability Goals
- Module 13: Designing for High Availability Goals
- Module 14: Optimizing Performance of Deployed Mule Applications
- Module 15: Designing Secure Mule Applications and Deployments
- Module 16: Securing Network Communications between Mule Applications
- Module 17: Documenting Integration Solutions Architectures

Approximate agenda



- Day 1: Part 1 (Module 1 - Module 3)
- Day 2: Part 1 (Module 4 - Module 6)
- Day 3: Part 2 (Module 7 - Module 9)
- Day 4: Part 3 (Module 10 - Module 14)
- Day 5: Part 3 (Module 15 - Module 17)

How the course will work



- Is **case-study driven**
 - Everyone will use the same case study
 - Role play that we all work for **AnyAirlines**
- As the course progresses, different parts of an **integration solution architecture** for a strategic change initiative will be documented
 - Starting with a provided integration architecture template
- **Design tradeoffs** will be evaluated and whenever possible a **best choice** will be decided within the **context** of the enterprise and the particular use case
 - Each student's individual solutions will be compared and discussed as a group
- Light on Business Architecture, heavy on **Application** and **Technology** Architecture

How each module will work



- Lectures (slides)
 - To provide background **knowledge** and seed group discussions
- Reflection questions
 - **Group discussions** to reinforce **knowledge** and compare and contrast options and best practices
- Exercises
 - Build your **skills** by applying the **knowledge** presented in the slides
 - Provide **hands-on experience** creating an **integration architecture document** for the course case study

19

How each exercise will work



- Usually start with a **group discussion** followed by **individual work** to fill in specific parts of the document using **best practices** and **informed decisions**
- You will create design diagrams
 - The course materials are built using LucidCharts www.lucidchart.com
 - End with individual presentations and group discussions
- Some solutions are provided
- There are some **light coding** and **runtime operations** activities
 - Common development, deployment, and management tools and services are either **demonstrated** or you use them in the **hands-on exercises**

20

- Available on MuleSoft Learning Management System
 - <http://training.mulesoft.com/login.html>
- **Student files** (ZIP)
 - Starting files and documents needed to complete some of the exercises
 - Exercise solutions
 - Reference materials
- **Course slides** (ZIP of PDFs)

At the end of this course, you should get certified!

- After you learn & master the content in this course, get the **MuleSoft Certified Integration Architect – Level 1** certification!
- This class comes with a **voucher for 2 attempts** for the exam
 - You will receive an email on the last day of class instructions to take the exam and a voucher code

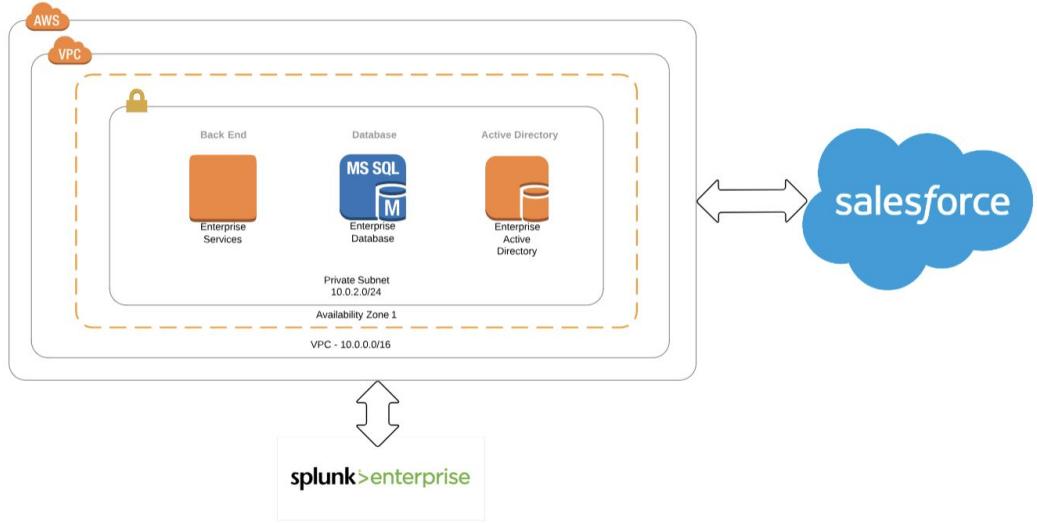


- Sign up for a trial account at <http://anypoint.mulesoft.com>
- Download and install Anypoint Studio 7 from <https://www.mulesoft.com/lp/dl/studio>
- Either have Microsoft Word installed on your computer, or have access to a Google Docs account
- Have software to draw architecture diagrams, such as
 - <https://www.yworks.com/products/yed>
 - <https://www.archimatetool.com/download/>

Introducing the course case study



The course case study

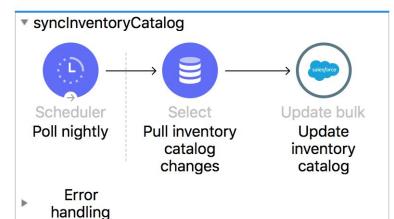
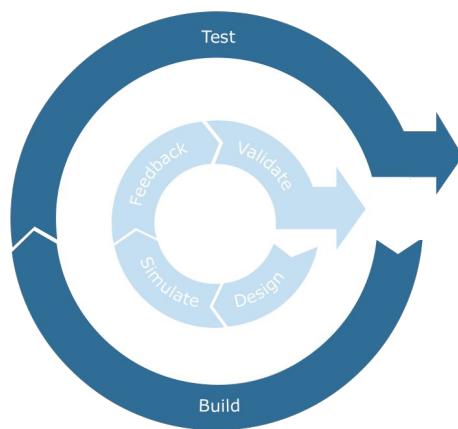




PART 1: Architecting and Designing Integration Solutions



Goal



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



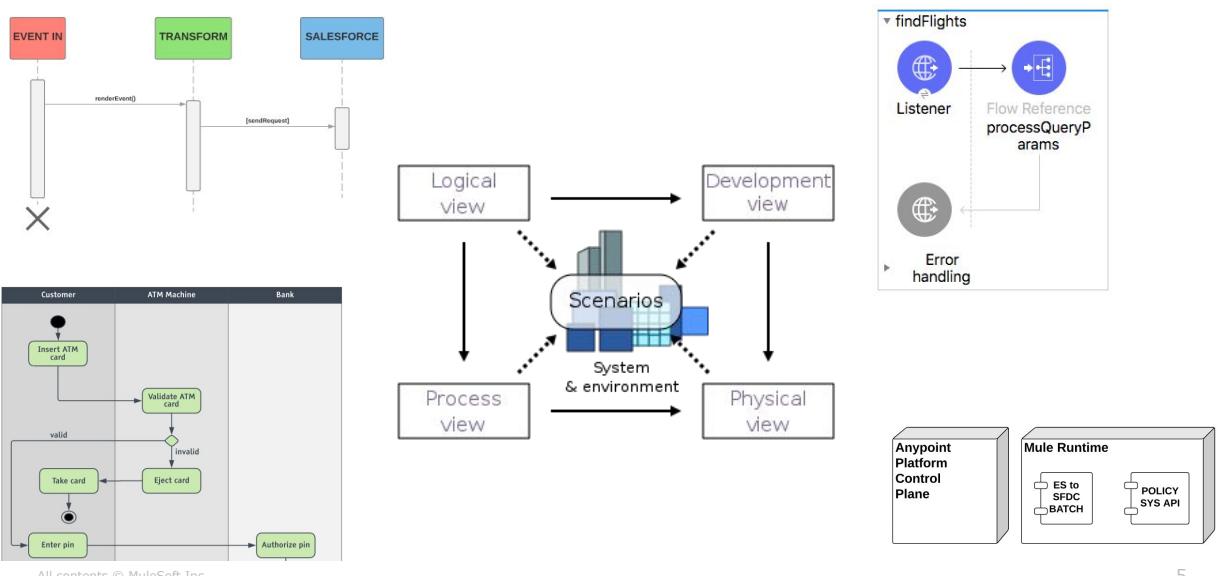
- Architect integration solutions
- Identify Anypoint Platform components and capabilities
- Design integration solutions using Mule applications
- Choose appropriate Mule processing models
- Choose Mule event transformation and routing patterns
- Design testing strategies for Mule applications



Module 1: Introducing Integration Solutions Architectures



Goal



All contents © MuleSoft Inc.

5

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



- Describe the objectives of enterprise integration solutions
- Summarize how to architect for customer success with Anypoint Platform
- Describe how integration solutions using Anypoint Platform are documented
- Start using an architecture template for the course case study

All contents © MuleSoft Inc.

6

Describing the objectives of enterprise integration solutions



What characterizes integration solutions across an enterprise?



- **Integration solutions** broadly involve linking together different computing systems and software applications physically or functionally, to act as a coordinated whole
- Particular challenges arise when integration efforts are carried out across internal and external **organizational boundaries**
 - Various stakeholders across the enterprise have **multiple**, and **sometimes conflicting, goals**
 - These stakeholders also have different **assumptions, understandings**, and **language** to describe these goals
 - Enterprise systems often have additional **requirements for reliability, availability, and performance**
- Integration solutions within an enterprise must address these enterprise-wide concerns

The most important objective of an integration solution architecture



- To address the **in-scope requirements** of **stakeholders** related to the integration scenarios and use cases
 - This requires documenting **different views** to address these **goals** and **concerns**
 - Additional view and documentation may be required to align **stakeholders** from **different business units or job roles**

Introducing the course case study



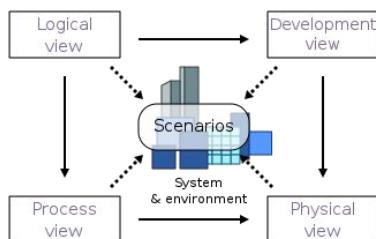
This class focuses on the design phase



- This class assumes the **analysis phase** steps (scenarios) have already been **completed** and agreed upon
 - This initial **project phase** includes important decisions and agreements, but it is mainly independent of MuleSoft tools or Anypoint Platform
 - The class case studies **already incorporate these decisions**

All contents © MuleSoft Inc.

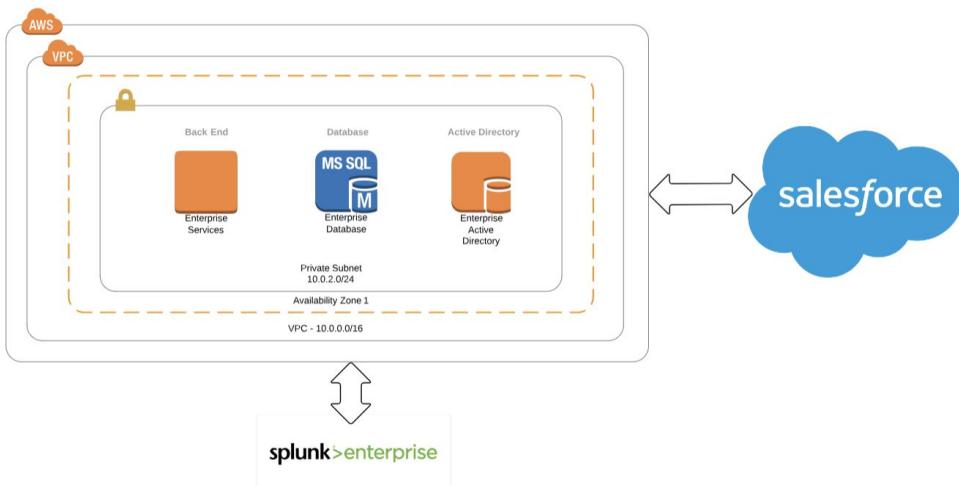
11



Case study



- Included in the course student files



All contents © MuleSoft Inc.

12

Identifying stakeholders involved in an integration project



Examples of stakeholders involved in integration solutions



Stakeholder type	Description and interests
Project sponsor	<ul style="list-style-type: none">Drives the project
Architects	<ul style="list-style-type: none">Responsible for implementing this and other integration solutions
Systems integrators and other external stakeholders	<ul style="list-style-type: none">Responsible for external systems that interact with the integration systems
Auditors	<ul style="list-style-type: none">Verify compliance, policies, and integrity of the companyOften can be disruptive
Users	<ul style="list-style-type: none">The business users (consumers) of the system

Responsibilities related to Anypoint Platform



Stakeholder type	Description and responsibilities
• Users	• The business users of the system
• Administrators	• Users that manage the system
• Developers and Architects	• Responsible for implementing this and other integration solutions
• Systems integrators and other external stakeholders	• Responsible for external systems that interact with the integration systems
• Analysts and Managers	• Responsible for managing the scope, business value, cost, resource usage, etc. of the project, programs, initiatives, etc.

Exercise 1-1: Identifying stakeholders and interests for an integration solution scenario



- Explore the course case study
- Describe the objectives of an integration solution scenario
- Identify stakeholders for an integration solutions scenario

An integration architecture must communicate with the various stakeholders



- An integration architecture must carefully **document** and **balance** the needs and **requirements** from all the different, and sometimes opposing, **stakeholders** and their **viewpoints**
- Different architectural **views** are constructed to build consensus with and between
 - Non-technical business level vs. technical level stakeholders
 - End users
 - Development vs. deployment vs. runtime operations stakeholders
 - Implementers vs. managers vs. executives

An integration architecture must communicate across contexts



- These different views communicate the intended architecture within **contexts** understood by each type of the stakeholders
 - This enables all stakeholders to **verify** and **accept** that the proposed systems will address their **requirements** and **concerns**
- Some architecture documents must translate or bridge between the contexts of different stakeholders
 - Business focused vs. technical focused stakeholders
 - Executive level stakeholders vs. managers vs. implementers

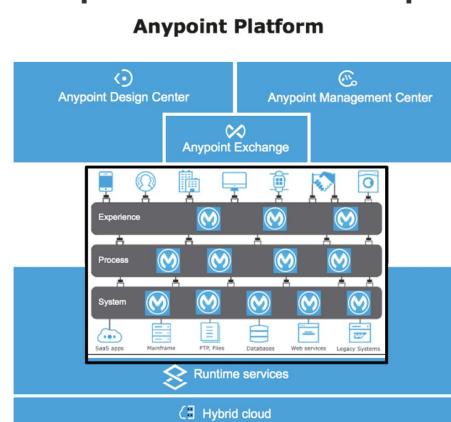
Architecting for customer success with Anypoint Platform



What is Anypoint Platform?



- A **unified**, highly productive, **hybrid** enterprise **integration platform** that
 - Creates a seamless collection of **integration applications**
 - Manages their **complete software development lifecycles**



MuleSoft's point of view on integration solution architectures vs. enterprise architectures



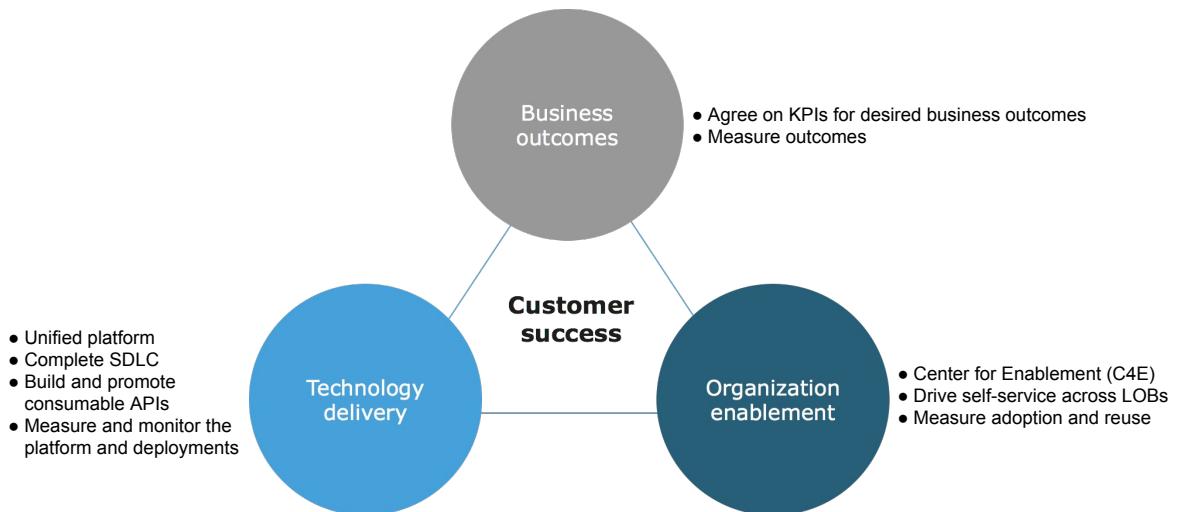
- MuleSoft generally divides architecture projects into two broad areas
- **Integration solutions architecture** (covered in this course)
 - Documents individual integration initiatives
 - For example message-based integration, batch processing, ETL, often in the form of direct integration between two systems
- **Enterprise architecture**
 - Continually builds up an organization's **application network**
 - Covered in the **Anypoint Platform Architecture: Platform Architecture course** and the corresponding **MuleSoft Certified Platform Architect** exam
 - Often involves **layering, reusing, and combining API-Led** initiatives **across organization boundaries**
 - Provides platforms and frameworks to guide individual solution architectures

MuleSoft's point of view on integration solutions



- For both types of architectures, MuleSoft has a strong **point of view** on how to effectively deliver business value across the enterprise using **integration solutions**
 - This includes **proven technology, delivery frameworks, and architectural styles**, focused on customer outcomes
 - Many MuleSoft customers are moving towards building up an application network while still moving fast towards delivering critical individual integration solutions
- MuleSoft can help you decide when and how to build up an enterprise architecture and application network along with your ongoing integration solutions

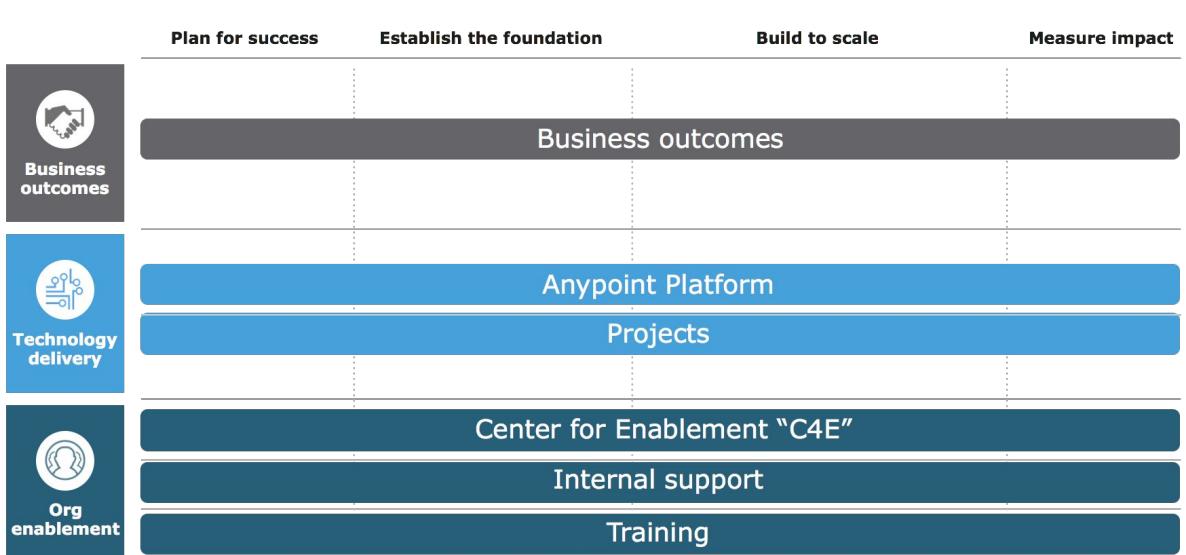
The three aspects of MuleSoft's **outcome based delivery** (OBD) approach



All contents © MuleSoft Inc.

24

Outcome based delivery timelines



All contents © MuleSoft Inc.

25

MuleSoft has a blueprint for you to follow



	Plan for success	Establish the foundation	Build to scale	Measure impact
Business outcomes	Agree on business outcomes and KPIs Develop the overall success plan	Monitor and manage	Refresh the success plan	Measure business outcomes
Technology delivery	Define Anypoint platform vision and roadmap Design Anypoint platform architecture and implementation plan	Deploy Anypoint Platform	Refine and scale Anypoint Platform	Measure Anypoint platform KPIs
	Prioritize IT projects and quick wins Staff and onboard the project teams	Define reference architecture Launch initial projects and quick wins	Onboard additional project teams Launch additional projects	Measure project KPIs
Org enablement	Assess integration capabilities Establish the C4E operating model	Build and publish foundational assets Evangelize	Drive consumption	Measure C4E KPIs
	Onboard MuleSoft Determine the internal support operating model	Staff, train and launch team Publish support guidance and self-serve materials	Monitor Anypoint Platform	Measure support KPIs
	Agree on initial roles Train the initial team(s)	Develop the broader training plan Launch experiential learning opportunities	Update training plan	Conduct skills assessment

Note: For details about individual steps, contact your MuleSoft Customer Success Manager

26

Documenting integration solutions



Identifying the types of audiences addressed in an integration solution architecture



- The developers
- The operations staff
- Business stakeholders

Identifying the types of documentation targeted to **developers** in an integration solution architecture



- Required use cases
- Logical views of systems, sub-systems
- Logical views of data and interfaces
- Non-functional requirements (NFRs) and service level agreements (SLAs)
- Process views of interactions and design decisions
- Key decisions, requirements, and tradeoffs
- Part 1 of the class focuses on these views and documents

Identifying the types of documentation targeted to **ops** in an integration solution architecture



- Development view of projects
- Physical view of systems and networks
- Part 2 and 3 of the class focuses on these views and documents

Documenting use cases for integration solutions



- **User stories** are often used in more modern Agile development, and have a specific format
- They are place keepers to begin conversations about requirements
 - **Use cases** are another option, but do not have a specific format

User story template:

- **As a** <what type of user>
- **I want/need to** <goal>
- **So that I can** <business objective>

Example:

- **As a** Market Data Analyst
- **I need to** run the Salesforce & Google analytics reports
- **So that I can** build the monthly media campaign plans

User stories need detailed acceptance criteria

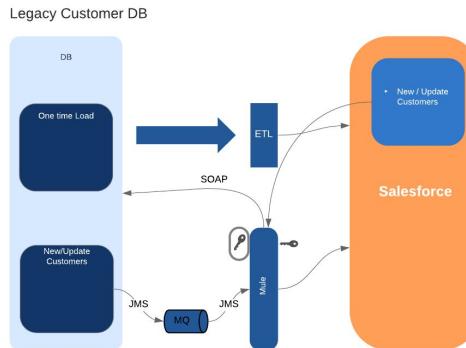
- **Acceptance criteria** communicate to the development team how to decide when the user story is considered completed (delivered)
- This helps limit the project scope and set expectations

User story	Acceptance criteria
<ul style="list-style-type: none">● As a Market Data Analyst● I need to run the Salesforce & Google analytics reports● So that I can build the monthly media campaign plans	<p>Ensure the Marketing Data Analyst is able to:</p> <ul style="list-style-type: none">● Access the Salesforce & Google Analytics reports● Create the monthly media campaign plan for a specified region (e.g. Region 9)● Access a Contacts list● Email the prepared monthly media campaign to one or more selected contact(s)

Use case realizations



- It can be helpful to collect the user stories together into an **integration stories map** to show the information flow between all the user stories
- This is baseline project milestone to scope the project so development can proceed



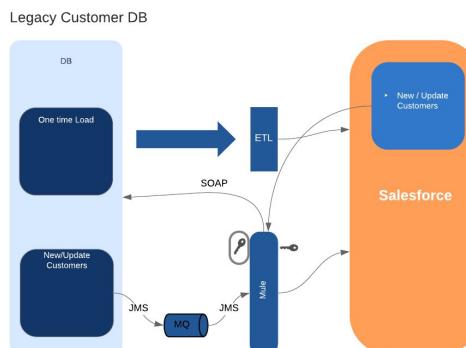
All contents © MuleSoft Inc.

34

Exercise 1-2: Document user stories for an integration solution



- Identify user stories for a case study
- Collect user stories together into an integration stories map



All contents © MuleSoft Inc.

35

Documenting requirements



Requirements are often discovered from user stories



- Functional requirements are often discovered from user stories
 - Triggering events
 - Acceptance criteria or other expected outcomes
 - Expected errors and error handling
- Non-functional requirements may exist beyond or outside a user story
 - May be invented by industry or other external authorities
 - Are constraints on other requirements
 - "Single sign-on is centrally supported by the enterprise LDAP server"
 - "The system must survive Mule runtime restarts"
 - "If the system is offline for 30 seconds, a status page must appear"
 - Requires defining failure, "up", "down", time windows, etc.

Identifying the types of documentation in an integration solution architecture

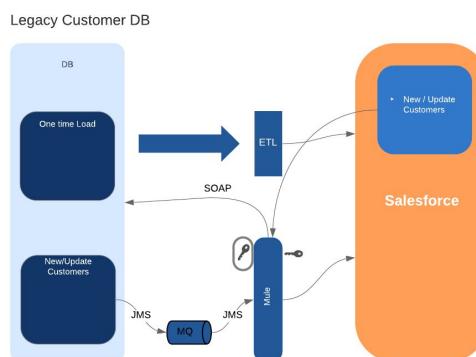


- Required use cases
- Views of systems, sub-systems
- Views of data and interfaces
- Functional requirements (FRs), non-functional requirements (NFRs), and service level agreements (SLAs)
 - **NFRs** must carefully specify criteria and tolerances
 - For example: "All data at rest must be encrypted per corporate standards"
- Views of interactions and design decisions
- Key decisions, requirements, and tradeoffs

Exercise 1-3: Document requirements



- Identify functional requirements for use cases
- Collect and identify non-functional requirements for an integration solution



Documenting interactions



How to identify and document interactions

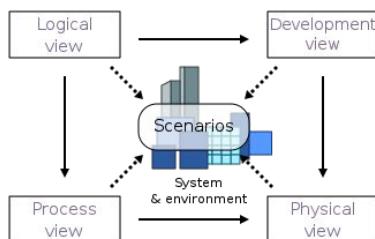


- For each use case, the details of all the interactions between systems and stakeholders must be defined
- To work efficiently, use cases must be prioritized and the details specified first for the most important use case(s)

The **4+1** views model is one common approach to illustrate views for software intensive systems



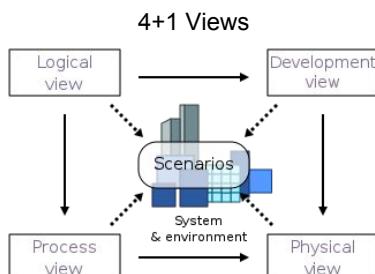
- Standard templates and best practices should be agreed upon and used for architecture documents
- The **4+1** views describe software systems
 - From the viewpoint of different stakeholders
 - Such as end-users, developers, and project managers
 - At various technical levels and different phases of a project's lifecycle
- https://en.wikipedia.org/wiki/4%2B1_architectural_view_model



The architecture methodology used in this course



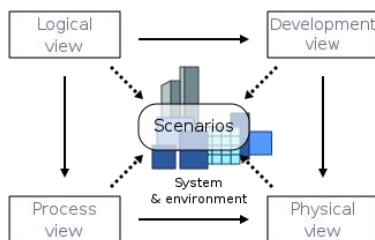
- In the course exercises, you will be filling in parts an architecture specification
- There are many widely accepted architectural approaches and styles
- In this course, the architecture templates are based on **4+1 views**



The 1 in the 4+1 view model is a set of **scenarios**



- The scenarios illustrate the behavior of the system from an **end user** perspective
 - Helps to **validate** the **architectural design** and **requirements** with key stakeholders
 - In typical integration solutions, the end user might not be a person
 - May be another system, sub-system, scheduler, or cron job



Reflection questions



- Have you used 4+1 views before?
- What other architecture approaches have you used?
- Have you used user stories or use cases, and if so, how did they compare with the user stories template presented earlier?

Architecting with 4+1 views



Defining architectural views for Mule applications

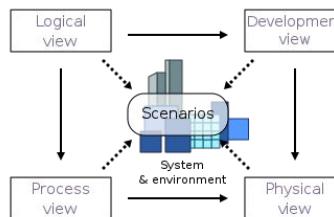


- Combining various architectural views related to a Mule application provides a complete realization of an integration solution using MuleSoft
 - Often use various UML diagrams
- Sequence** and/ **activity** diagrams are typically associated with use case realizations of the systems
 - Mule flows are similar to activity diagrams, so mocked Mule flows can also be used to help design integration solutions
- Later in the project lifecycle, **deployment diagrams** can be associated with use case realizations
- Per use case, architects may provide other views at various detail levels (zooming in or out) to help clarify the integration solution

How 4+1 views are tied to common project lifecycle phases



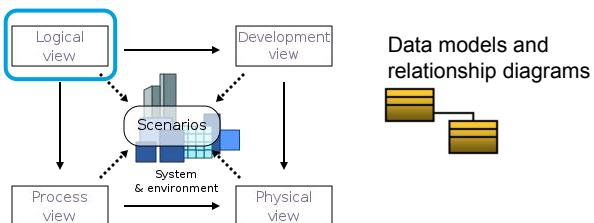
- The analysis phases focuses more on the business requirements
 - Usually focuses on the **scenarios** and **logical views**
- The design phases refines the logical views into more actionable views
 - Oriented to more technical stakeholders
 - Design actual interfaces and interactions to realize the user stories
 - **Process views, development views, and physical views**
- The rest of the project lifecycle (implementation, deployment, maintenance) is informed by the design phase
 - Earlier phases are continuously revisited and refined in an **iterative manner**



The 4+1 **logical view** illustrates end-to-end system functionality



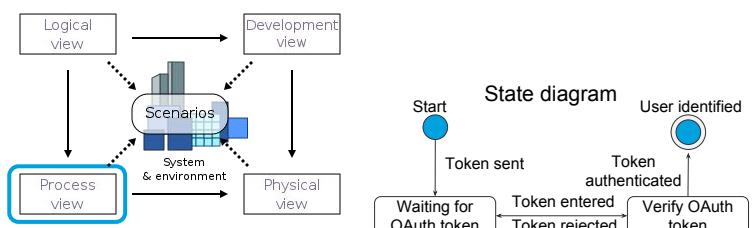
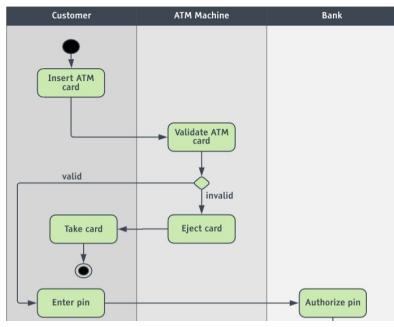
- Usually defines and documents **system, stakeholders, and interfaces**, and their **relationships**
 - Major big picture view
 - Diagrams show the **relationships** between systems and stakeholders and existing relationships and connectivity
 - Includes **data models** for the integration domain and systems of record/reference
 - These views are included in the larger enterprise distributed architecture



The 4+1 process view illustrates the runtime behavior of systems



- Give more detail of a particular movement of information, usually per use case
- Specifies how SLAs and NFRs are met, including security policies
- Documents both success and failure paths
- Documents human interactions in the business processes



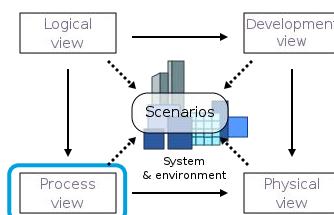
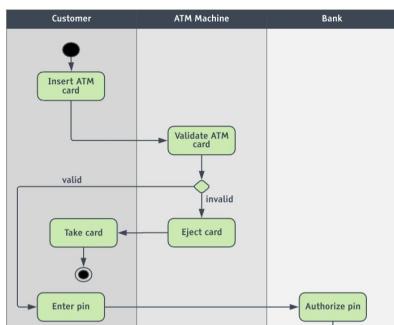
All contents © MuleSoft Inc.

51

The 4+1 process view illustrates the runtime behavior of systems



- Activity diagrams illustrate communication across **swimlanes** at a more technical level, while still skipping implementation details
 - Swimlanes visually partition ownership by person, group, system, or sub-system
 - Document **concurrency, distribution, integrators, performance, and scalability** of systems and processes

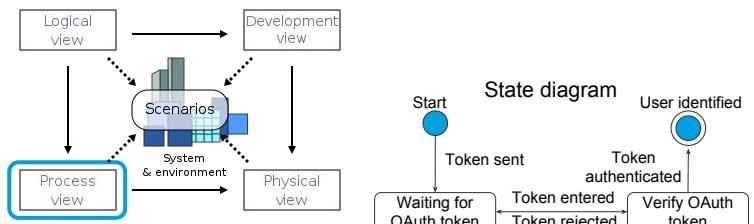
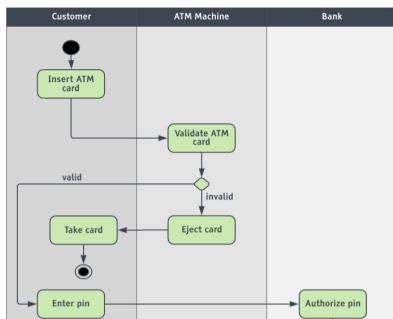


52

Anypoint Studio can quickly represent process views



- Flows are similar to activity diagrams
- They do not need class diagrams or state diagrams



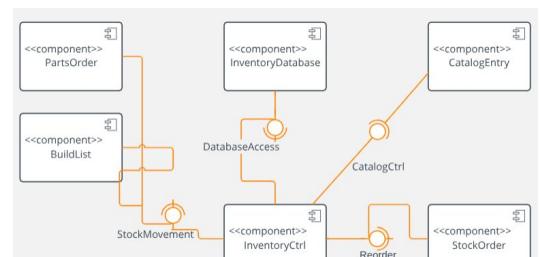
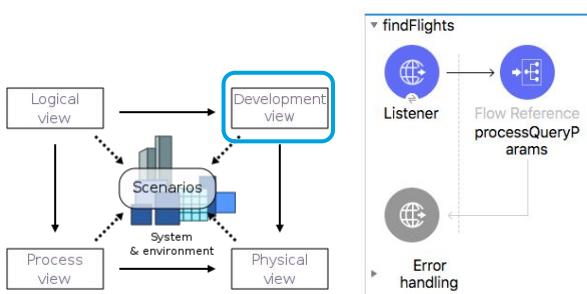
All contents © MuleSoft Inc.

53

The 4+1 **development view** illustrates systems from a developer perspective



- Documents **component** and **packages** with diagrams to illustrate how to code the solution
- Often the development view does not directly relate to Mule apps
 - MuleSoft's drag-and-drop development tools often shield the developer from these coding details
 - But they may be included in the larger enterprise distributed architecture



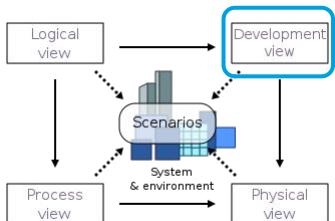
All contents © MuleSoft Inc.

54

The 4+1 **development view** categories



- Development processes to deliver solutions
 - Test automation and strategies, performance testing, release management, branch/merge strategies, Continuous Integration (CI)
- Project scoping and reuse
 - Microservice architecture, API-Led, reusable libraries, version and platform dependencies and decisions
- Solution quality
 - Standards and frameworks, protocols, logging, and instrumentation



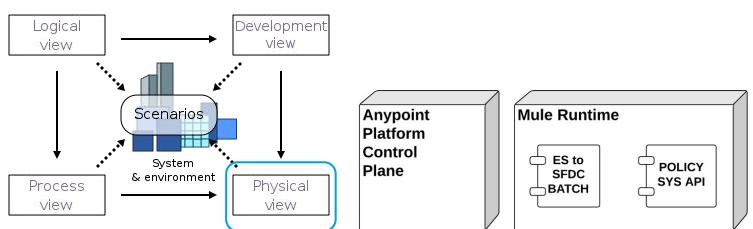
All contents © MuleSoft Inc.

55

The 4+1 **physical view** documents systems deployments for system integrators



- Uses **deployment diagrams** to illustrate the physical view of systems, including showing
 - Components ("nodes") that currently exist or will exist
 - For example enterprise databases, JMS servers, or SaaS systems (like Salesforce)
 - What software components ("artifacts") run on each node
 - For example Mule applications, custom Java code, or Apex APIs
 - How the different pieces are connected (e.g. JDBC, REST, SOAP)



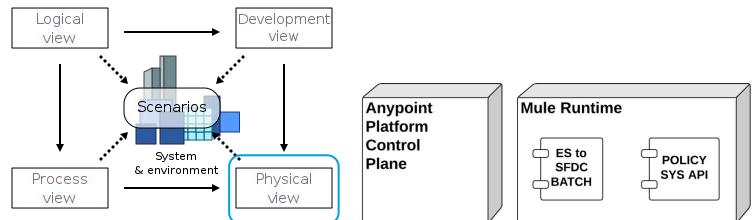
All contents © MuleSoft Inc.

56

Common 4+1 **physical view** documents



- High level network topology
 - Database instances, Mule runtimes, CloudHub workers, DLBs, firewalls, VPCs and network tunnels
- Licensing
- Infrastructure, including runtime and control planes
- Continuous Delivery (CD) and other environments
 - Mule environments must coexist and conform with existing environments



All contents © MuleSoft Inc.

57

Other required documentation



- **Maintenance, operations, and security** may need specialized documents
- These stakeholders may need separate documents and manuals culled and refined from the 4+1 views

All contents © MuleSoft Inc.

58

Exercise 1-4: Document 4+1 views for a use case



- Add 4+1 view sections to an architecture document
- Locate 4+1 views for an existing MuleSoft-enabled solution
- Create a process view for a use case
- Explore completed 4+1 views for a use case

Reflection questions



- Which 4+1 views often use use case diagrams or user stories?
- Which 4+1 views often use sequence diagrams?
- Which 4+1 views often use activity diagrams?
- Which 4+1 views often use class diagrams?
- Which 4+1 views often use state diagrams?
- What is the difference between a state diagram and an activity diagram?

Exercise steps



- Identify uses cases and requirements
 - Open the case study for the course
 - Discuss the use cases to be implemented by this integration project
 - Identify and discuss functional requirements
 - Identify and discuss non-functional requirements
 - Identify deliverable documentation that will explains how all requirements of the case study will be realized

Exercise steps



- Begin to document a use case for the integration solution
 - Open the system integration architecture document template
 - Add the case study description and associated use cases to the architecture template
 - Add functional requirements to the architecture document
 - Add non-functional requirements to the architecture document
 - Identify the types of views required to complete the architecture document
 - Identify the types of optional views that may also be included to complete the architecture document

- Have each student present their solution, with group discussion guided by the instructor
- Compare and discuss the different solutions presented
- Fill in the solution architecture document with an agreed solution

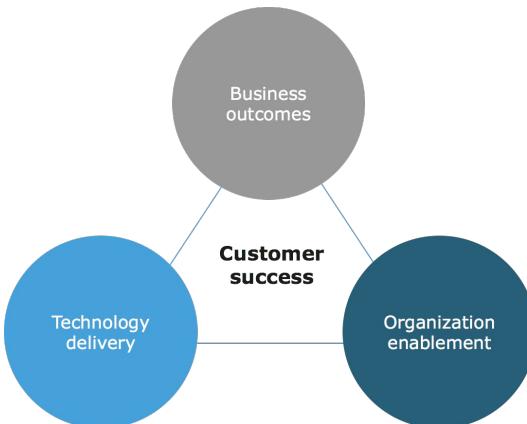
Summary



Summary



- MuleSoft uses an outcome based delivery approach to deliver integration solutions



All contents © MuleSoft Inc.

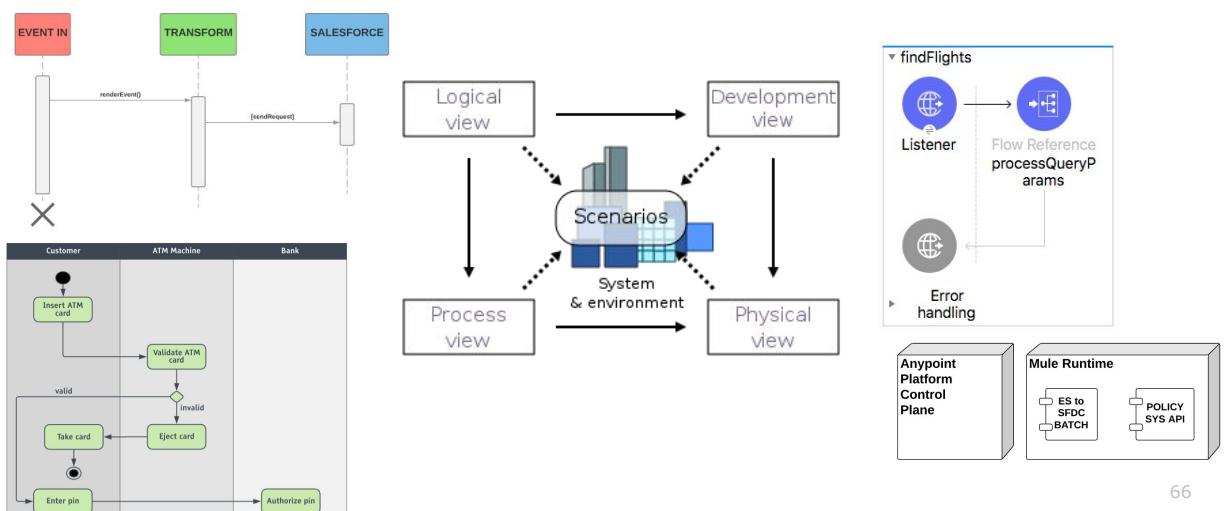
10

65

Summary



- MuleSoft integration architectures document various views

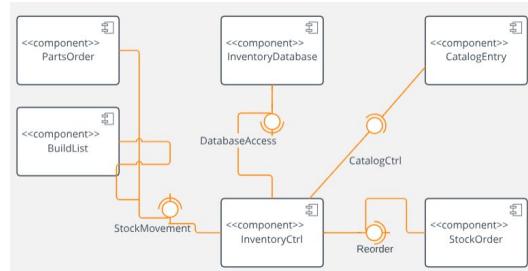


66

Summary



- The development views can be simplified by using the MuleSoft toolset
 - May not need to create class and state diagrams for the logical view
 - May not need to create component and package diagrams for the development view
- MuleSoft has a proven point of view on how to successfully architect integration solutions





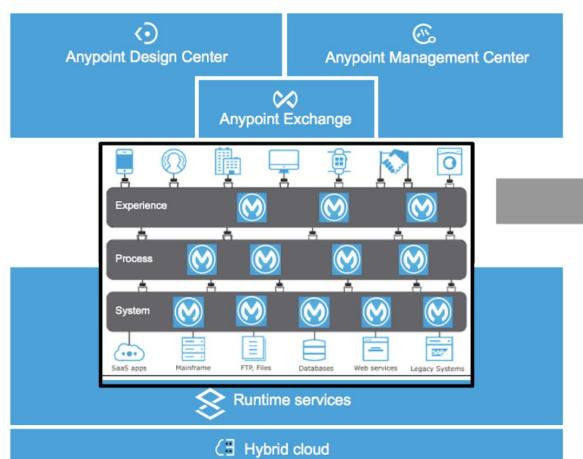
Module 2: Identifying Anypoint Platform Components and Capabilities



Goal



Anypoint Platform



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



- Identify overall design intentions of Anypoint Platform
- Review Anypoint Platform capabilities and high-level components
- Distinguish between Anypoint Platform service and deployment models
- Align Anypoint Platform components and capabilities with an integration use case

Putting Anypoint Platform and Mule applications into an integration architecture



- Before creating an integration architecture for the course case study, you must understand the **platform** and **tools** provided by **MuleSoft**
- This is the goal of the next two modules, before starting to fill in the architecture documents in the rest of the course
 - Module 2: Identifying the Components and Capabilities of Anypoint Platform
 - Module 3: Designing Integration Solutions with Mule Applications
 - Identify the components and capabilities of the Mule runtime and associated development toolsets

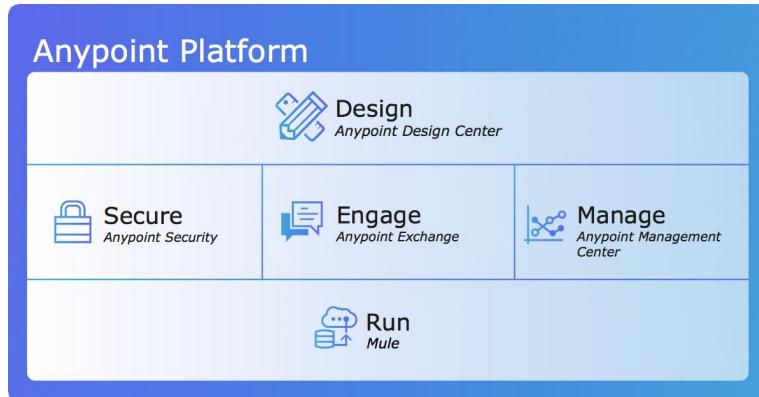
Introducing Anypoint Platform



What is Anypoint Platform?

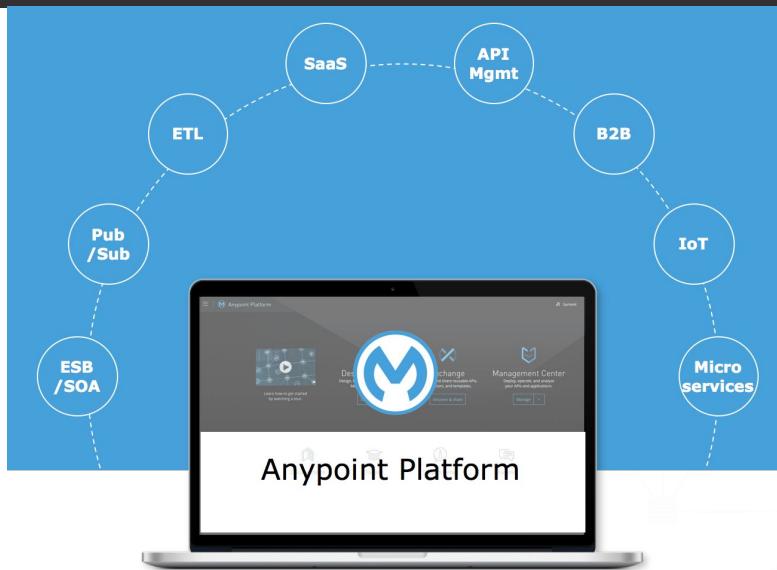


- A **unified**, highly productive, **hybrid** integration platform that creates a seamless distributed system of apps, data, and devices
- Can also manage full API lifecycles to promote API-led development



7

Anypoint Platform manages Mule application lifecycles



Advanced enterprise platform for designing, developing, and managing APIs and integrations

- Uniquely built as a single product
- Deploy anywhere
- Flexible and wide range of use cases

All contents © MuleSoft Inc.

9

One unified platform to design and manage integration solutions, and exchange related assets



Specialists



Admin, Ops,
DevOps



Ad-hoc
integrators



App devs



Rapid
development

Design Center



Collaboration
and self-service

Anypoint Exchange



Visibility
and Control

Anypoint Monitoring and Visualizer



Lean runtime
Mule runtime

10

One common runtime for all types of deployments



Specialists



Admin, Ops,
DevOps



Ad-hoc
integrators



App devs



Rapid
development

Design Center



Collaboration
and self-service

Anypoint Exchange



Visibility
and Control

Anypoint Monitoring and Visualizer



Lean runtime
Mule runtime



On-premises &
Private Cloud



Hybrid



Hosted by
MuleSoft

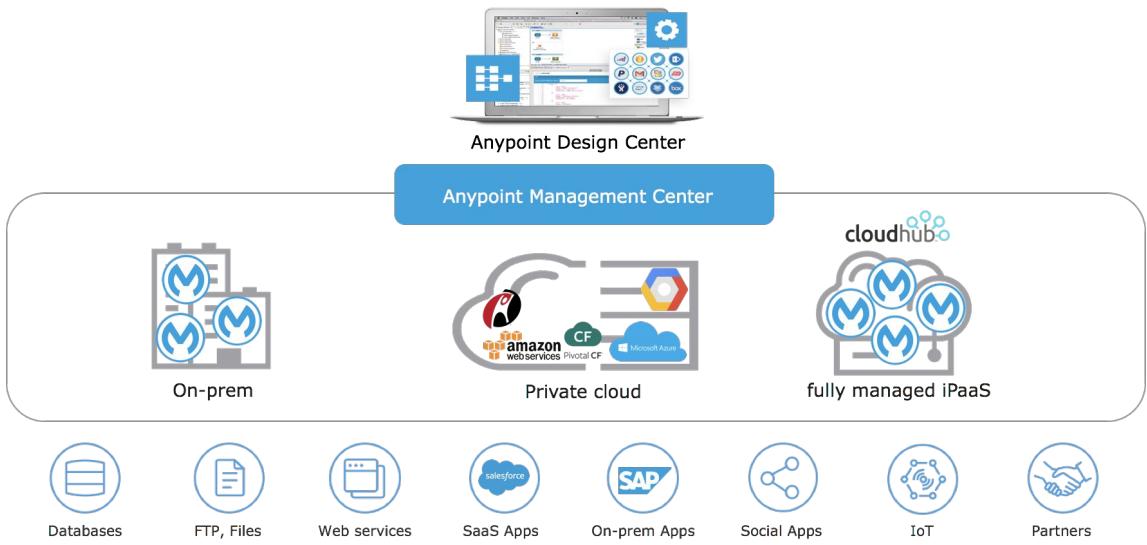


Cloud service providers



11

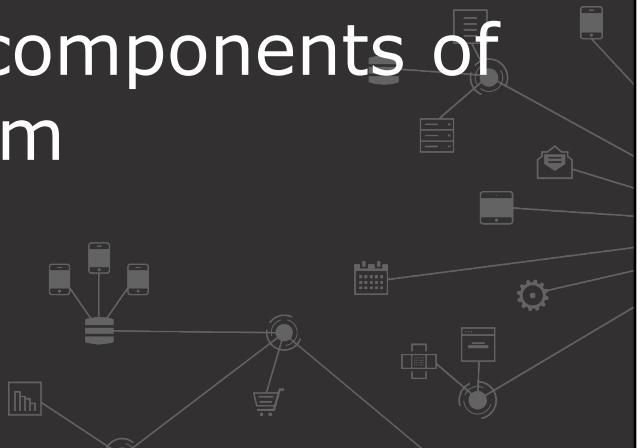
Design once, deploy anywhere



All contents © MuleSoft Inc.

12

Identifying the components of Anypoint Platform



Anypoint Platform is a collection of runtimes, frameworks, tools, and web applications

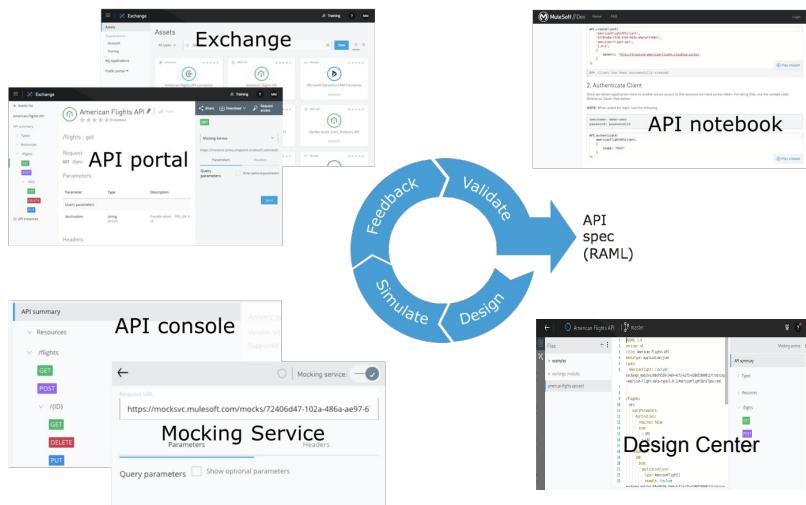


- **Tools and frameworks** for building applications
- One **Mule runtime** for running Mule applications and applying policies
 - The same Mule runtime is used in MuleSoft-hosted infrastructure (CloudHub) or in customer-hosted infrastructure (on-premises or in the cloud)
- A suite of **web applications** for
 - Discovering and learning about APIs and other assets
 - Building integration applications that consume APIs
 - Deploying, running, managing, and monitoring applications
 - Defining and managing APIs

Anypoint Platform components used to develop and manage APIs during the MuleSoft design phase



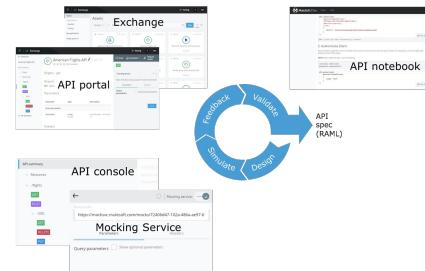
- A unified platform to design, build, test, share, and manage APIs



Implementing with an API-led approach using the unified Anypoint Platform



- **Mule applications** can first be designed with API specifications that are easier for less technical stakeholders to understand
 - Write, publish, and version APIs in **Anypoint Design Center**
 - Manage APIs with **Anypoint API Manager**
 - Offload API governance and policies from the API implementation (the Mule application) to a centralized management plane used by runtime admins, not developers
 - Share, mock, test, and reuse APIs with **Anypoint Exchange**



All contents © MuleSoft Inc.

29

MuleSoft recommended REST API specification options



- MuleSoft recommends using a modern, open, flexible API documentation language to model REST services
 - Should be **language agnostic** to easily model XML, JSON, and Java objects in a more readable syntax
 - Should be readable by less technical, more business focused staff
 - Should still allow auto-generation of skeleton implementations by tools
- **REST API Modeling Language (RAML)** is a MuleSoft invented open standard
 - Based on YAML (YAML Ain't Markup Language)
- **OpenAPI Specification (OAS)**
 - Formerly called Swagger
 - Another open standard to define REST interfaces in YAML or JSON format

```
%%RAML 1.0
title: Orders API

/orders:
  get:
    /{orderId}:
      post:
        body:
          application/json
        responses:
          200:
          404:
```

All contents © MuleSoft Inc.

30

MuleSoft development tools and the Anypoint Platform fully support RAML and OAS



- Supports a **design-first approach**
- Interactions can first be architected and designed using REST API specifications
 - Typically with RAML or OAS
- Anypoint Design Center assists designing, sharing, versioning, and iterating on RAML and OAS specifications
- Based on a RAML or OAS specification
 - Anypoint Studio can **auto-generate skeleton implementation** flows using the APIkit component
 - Anypoint Connect can automatically create and publish an Anypoint Connector to Anypoint Exchange

Deploying, managing, and monitoring Mule applications



- Mule applications are deployed to a **Mule runtime**
 - Mule runtimes can be MuleSoft-hosted in the **cloud (CloudHub)** or **customer-hosted** in the cloud or on-premises
- A Mule runtime is a **lightweight Java-based integration platform**
 - Allows developers to connect apps together quickly and easily, enabling them to exchange data
 - Acts as a transit system for carrying data between apps (the Mule)
 - Can connect any systems using any protocols
 - Including HTTP, web services, JDBC, FTP, and JMS

Exercise 2-1: Explore Anypoint Platform and Anypoint Exchange



- Identify asset types supported by Anypoint Exchange
- Identify resources defined in a REST API
- Identify API dependencies and RAML fragments
- Identify how different versions of an API are stored in Anypoint Exchange

The screenshot shows the Anypoint Exchange web interface. At the top, there's a navigation bar with links for 'Secure | https://anypoint.mulesoft.com/exchange/' (highlighted), 'Apps', 'Aloha', 'V4 API - Integratio...', 'Exchange' (selected), 'MuleSoft Inc', a question mark icon, and 'SN'. On the left, a sidebar has 'Assets' selected, along with 'Organizations' (MuleSoft, MuleSoft Inc), 'My applications', 'Public portal', and 'Settings'. The main area is titled 'Assets' with a 'Search' input field. It shows four items: 'Salesforce Connector - Mule 4' (Connector, 5 stars, MuleSoft), 'Salesforce Connector for Einstein Analytics - Mule 4' (Connector, 5 stars, MuleSoft), 'Catalyst Accelerator for Healthcare' (Custom, 5 stars, MuleSoft), and 'Salesforce to Database Account Broadcast' (Template, 5 stars, MuleSoft). A 'New asset' button is at the bottom right.

33

Exercise step



- Log in to Anypoint Exchange using
<https://anypoint.mulesoft.com/login/#/signin?apintent=exchange>

Exercise step



- Review Anypoint Exchange asset types

The screenshot shows the Anypoint Exchange interface with the 'Assets' tab selected. On the left, a sidebar lists 'Organizations' (MuleSoft, MuleSoft Inc), 'My applications', 'Public portal', and 'Settings'. The main area is titled 'Assets' and shows a grid of four items:

- Connector**: Salesforce Connector - Mule 4 by MuleSoft (5 stars)
- Connector**: Salesforce Connector for Einstein Analytics - Mule 4 by MuleSoft (5 stars)
- Custom**: Catalyst Accelerator for Healthcare by MuleSoft (5 stars)
- Template**: Salesforce to Database Account Broadcast by MuleSoft (5 stars)

A 'New asset' button is located in the top right corner of the grid.

All contents © MuleSoft Inc.

35

Exercise solution



- Identify Anypoint Exchange asset types

The screenshot shows the Anypoint Exchange interface with the 'Assets' tab selected. On the left, a sidebar lists 'Organizations' (MuleSoft, MuleSoft Inc), 'My applications', 'Public portal', and 'Settings'. The main area shows a search result for 'flight':

"flight". Save this search

REST API ★★★★☆

Training: American Flights API by MuleSoft

A dropdown menu on the left shows categories: All types, Connectors, Templates, Examples, REST APIs, SOAP APIs, HTTP APIs, RAML fragments, and Custom. A search bar above the results contains the text "flight".

All contents © MuleSoft Inc.

36

Exercise step



- Locate assets in the public Anypoint Exchange for the American Flights API used in the Anypoint Platform Development: Fundamentals course

All contents © MuleSoft Inc.

37

Exercise solution



- Identify REST resources defined in a REST API

GET Get all flights
POST Add a flight
GET /{ID} Get a fil...
DELETE Delete...
PUT Update ...

All contents © MuleSoft Inc.

38

Exercise solution



• Identify API dependencies

The screenshot shows the 'Dependencies' section of the API Exchange for the 'Training: American Flights API'. It lists two dependencies:

- Training: American Flights Example 1.0.1**: RAML fragment
- Training: American Flight Data Type 1.0.1**: RAML fragment

Both dependencies are marked with a blue circular icon containing a white 'R'.

Version	Instances
1.0.1	Mocking Service
1.0.0	...

<https://api��点点.mulesoft.com/exchange/68ef9520-24e9-4cf2-b2f5-620025690913/training-american-flights-api/1.0.1/pages/home/>

39

Exercise solution



• Select an asset version

The screenshot shows the 'Asset versions for 1.0' section of the API Exchange for the 'Training: American Flights API'. It lists two asset versions:

Version	Instances
1.0.1	Mocking Service
1.0.0	...

A blue arrow points from the '1.0.1' row to the 'Mocking Service' instance.

<https://api��点点.mulesoft.com/exchange/68ef9520-24e9-4cf2-b2f5-620025690913/training-american-flights-api/1.0.1/pages/home/>

40

Exercise reflection questions

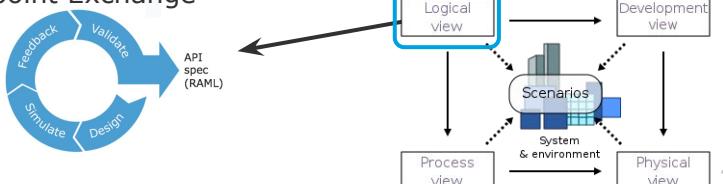


- If you ever designed or implemented a REST API
 - How did you document the REST API?
 - How RESTful was the API?
 - How easily could someone come in and start refactoring the REST API several years from now?

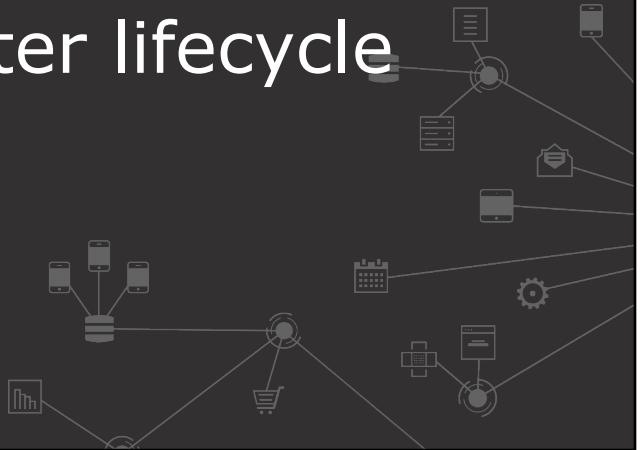
Some 4+1 views are created during the MuleSoft project design phase



- The 4+1 views drive the initial MuleSoft project design phase, including defining new APIs
- **The Anypoint Platform and Mule applications are an integral part of the design phase**
 - Can be used to **build proof of concepts** that can quickly mock the required user stories, in line with 4+1 views
 - You will use Anypoint Platform and development tools to mock some user stories
 - Often **without writing any code**
 - Custom components might be created or can be simulated with sample data and schema from Anypoint Exchange



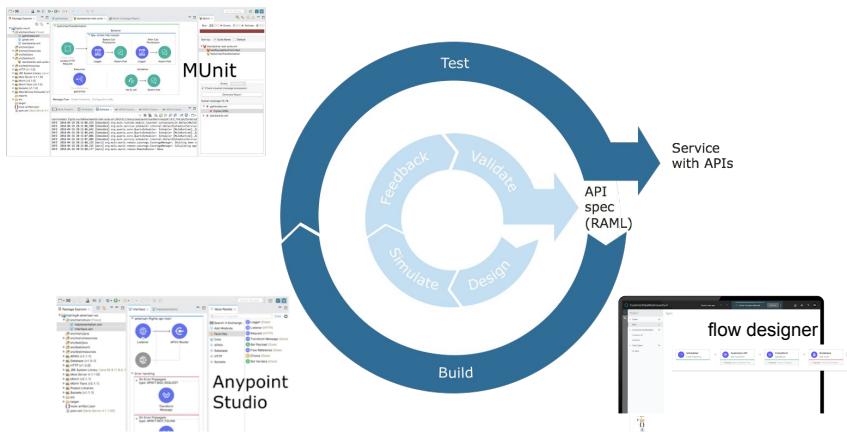
Designing for later lifecycle phases



Components used to **implement** an API or general Mule applications



- Anypoint Platform provides tools to implement and test Mule applications
- These tools can be used with or without API specifications

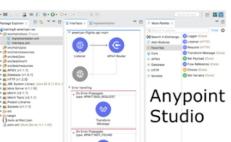


Both flow designer and Anypoint Studio create Mule applications



- **Mule applications** can be created in several ways
 - Visually using the online **flow designer** or locally using **Anypoint Studio**
 - Anypoint Studio provides some more advanced capabilities compared with flow designer
 - In a text editor by writing code (primarily XML) using Anypoint Studio (or other tools)
- Under the hood, Mule applications are **Java applications** (based on Java Spring) that are configured by Mule application XML files

```
<flow name="findFlights">
    <http:listener doc:name="Listener" config-ref="HTTP_Listener_config" path="flights" />
    <flow-ref doc:name="processQueryParams" name="processQueryParams"/>
</flow>
```



Anypoint Studio



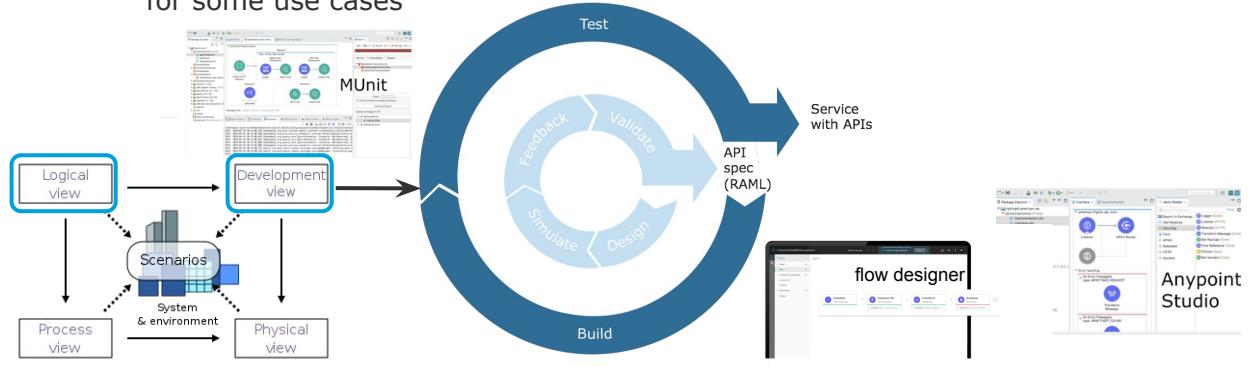
flow designer

45

4+1 views are elaborated during the MuleSoft project implementation and testing phases

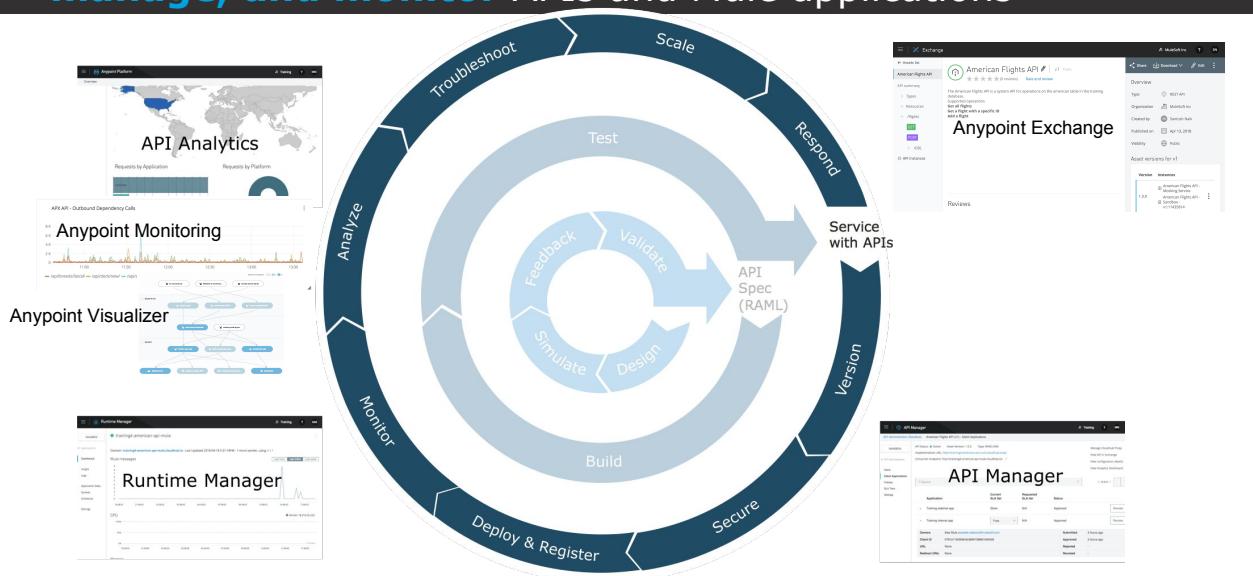


- Feedback from the design phase is used to refine the 4+1 views
- The completed architecture then drives **final implementation and testing**
 - You will use Anypoint Platform and development tools to fill in these views for some use cases



46

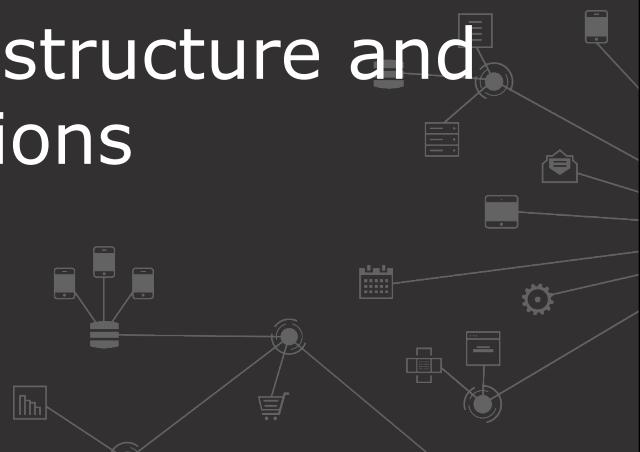
Anypoint Platform components used to **deploy, manage, and monitor** APIs and Mule applications



All contents © MuleSoft Inc.

47

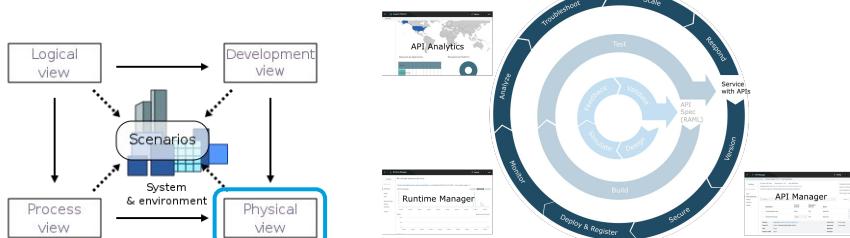
Identifying infrastructure and deployment options



Anypoint Platform deployment options

MuleSoft

- The same Mule application can be deployed to either **MuleSoft-hosted or customer-hosted** infrastructure
 - This infrastructure and related services are called the **runtime plane**
- In all cases, the Mule application is deployed to a Mule runtime
- The difference is in how the infrastructure is provisioned and managed to host the Mule runtime(s)



All contents © MuleSoft Inc.

49

Features supported by all runtime plane types

MuleSoft

- Deploying, stopping, starting, and restarting a Mule application through Runtime Manager
- Setting properties from Runtime Manager

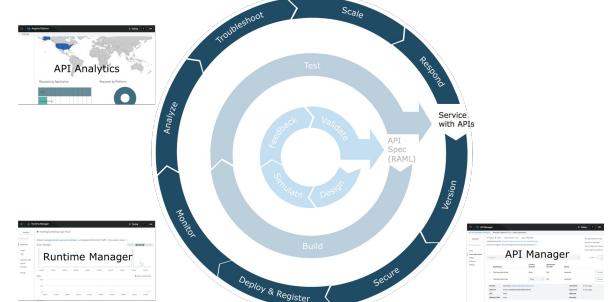
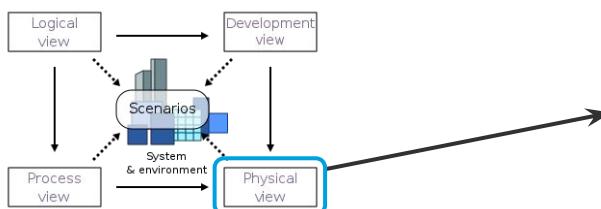
All contents © MuleSoft Inc.

50

Which 4+1 views are created to direct the deployment and maintenance phases



- The physical views are mainly used to design deployment and project maintenance of Mule applications and API implementations
 - The architecture shows **the process and the nodes** that run the process
 - In addition to the 4+1 view, the architecture can also document larger **CI/CD** processes or other automation
 - In this class, you will create all these architecture documents for the class use cases



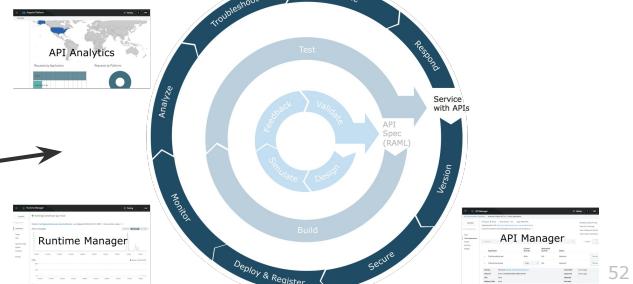
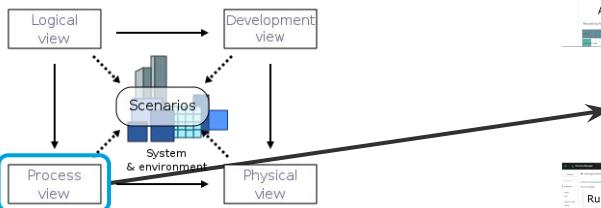
All contents © MuleSoft Inc.

51

Process views are also used to document distributed data exchange



- The **Process views** also supplement the physical views to decide on deployment infrastructure options
 - Activity diagrams document data flow across systems
 - The Process view communicates NFRs for distributed data exchanges, which then informs decisions in the Process views
 - Performance requirements and SLAs
 - Reliability and HA requirements and SLAs



All contents © MuleSoft Inc.

52

Applying Anypoint Platform components to the course case study



Applying Anypoint Platform to the course case study



- Now you can identify parts of the course case study that can benefit from Anypoint Platform components
 - Anypoint Studio or flow designer
 - API Manager
 - Anypoint Exchange
 - Runtime Manager
 - Anypoint Visualizer
 - Anypoint Monitoring



Exercise 2-2: Align Anypoint Platform components and capabilities with a use case



- Decide which Anypoint Platform components can be applied to meet the functional and non-functional requirements



Exercise step: Identify Anypoint Platform components to meet requirements



Requirement	Anypoint Platform Components	Comments

Exercise steps



- Open the course case study
- Identify which Anypoint Platform components can be applied to meet the functional and non-functional requirements
- Add preliminary sketches of Anypoint Platform components and capabilities to the architecture document

Exercise solution



- Open the solution architecture document from your student files
- Compare your architecture document with the provided solution architecture document

Summary



Summary



- A **unified**, highly productive, **hybrid** integration platform that creates a seamless distributed system of apps, data, and devices
- Can also manage full API lifecycles to promote API-led development



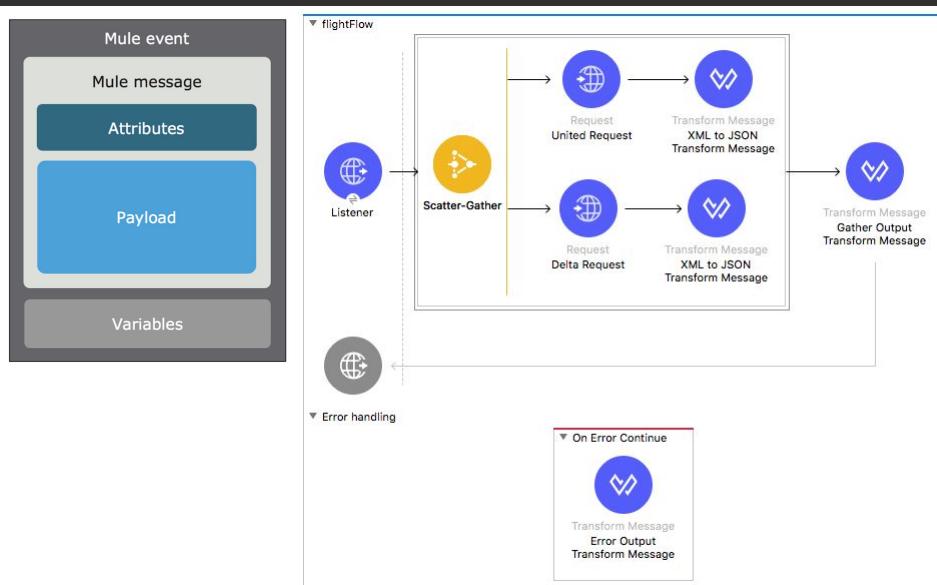
- Use **Anypoint Exchange** as a central repository for assets so they can be discovered and reused
 - Populate it with everything you need to build your integration projects.
- Use **flow designer or Anypoint Studio** to build integration applications
- **Mule runtimes** can be MuleSoft-hosted in the cloud (CloudHub) or customer-hosted in the cloud or on-prem



Module 3: Designing Integration Solutions with Mule Application Components



Goal



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



- Identify how Mule application components are typically used to implement integration solutions
- Identify how class loading isolation is implemented in Mule runtimes
- Apply Mule application components and capabilities to the course case study

Specifying detailed Mule application designs



- The last module identified at a high level how to leverage the unified Anypoint Platform framework and services to achieve use case requirements
- This module focuses on more specific details related to designing the implementations of specific use cases
 - Identify components that make up typical Mule application flows and their behaviors to
 - Exchange data with external resources using **Mule 4 connectors**
 - Transform data using **DataWeave 2**
 - Control the processing flow between **event processors**
 - Handle errors generated in flows
 - Troubleshoot **Mule 4 class loading issues**

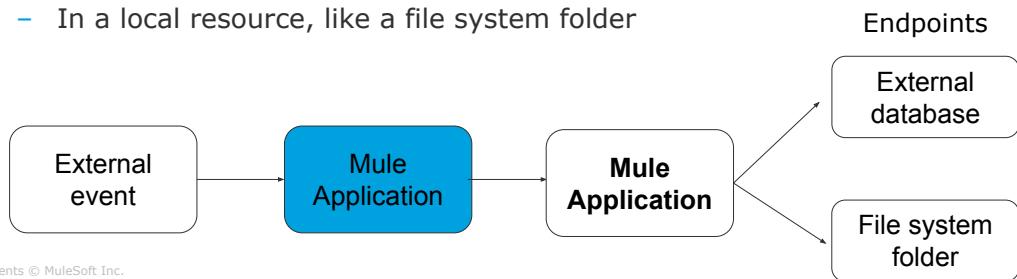
Identifying typical Mule Application components



What is a Mule application?



- A **Mule application** runs in a Java virtual machine (JVM) and is **triggered** by internal or external **events**, **processes** each event, and **routes** events to other **components** or **endpoints**
- The endpoints may be
 - In another Mule application
 - In another external system or resource
 - In a local resource, like a file system folder



All contents © MuleSoft Inc.

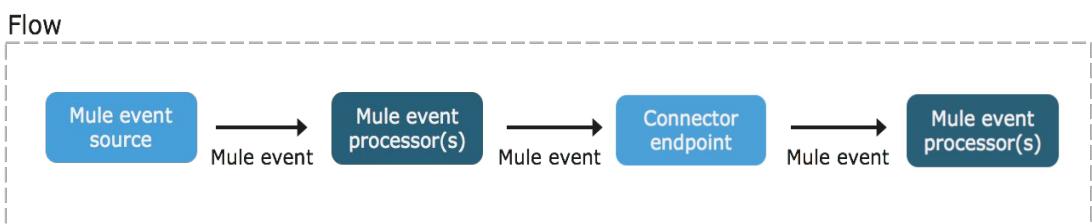
7

How Mule applications process events



- Mule applications accept and process **events** through a series of **Mule event processors** (also called **components**) linked together in one or more **flows**

Note: In Mule 3, components were called elements
- A flow is a linked sequence of event processors
 - Think of each **event processor** as a Lego block, while a **flow** is something you construct by linking the blocks together in a certain order



All contents © MuleSoft Inc.

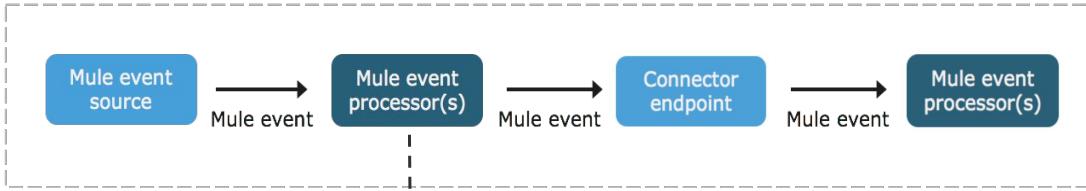
8

A Mule application consists of one or more flows

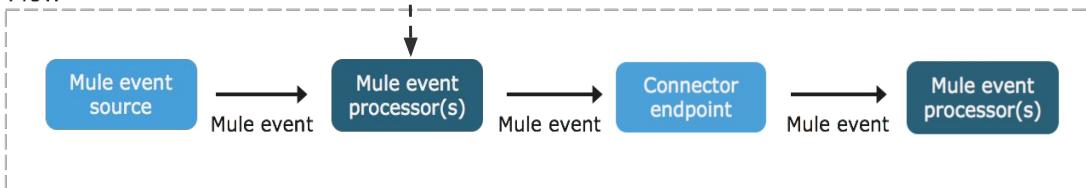


- Each flow can be independent of the other flows, or may invoke other flows in the Mule application

Flow



Flow



All contents © MuleSoft Inc.

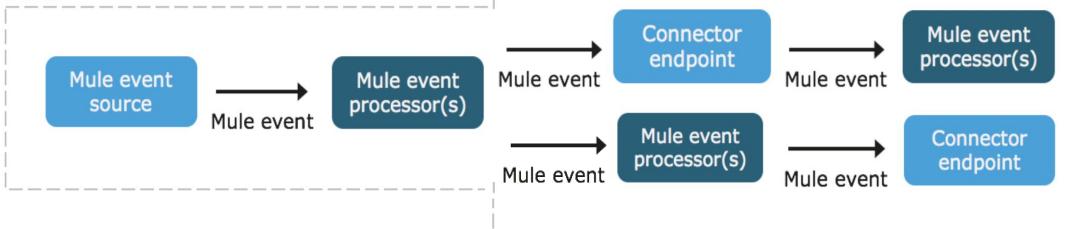
9

How Mule application flows are built and executed



- A flow can coordinate services within one or more flows
 - A Mule 4 flow will often execute one event in multiple different threads
 - Mule 4 optimizes asynchronous event processing to avoid some unnecessary thread switches
 - Can execute concurrent tasks in parallel

Flow



All contents © MuleSoft Inc.

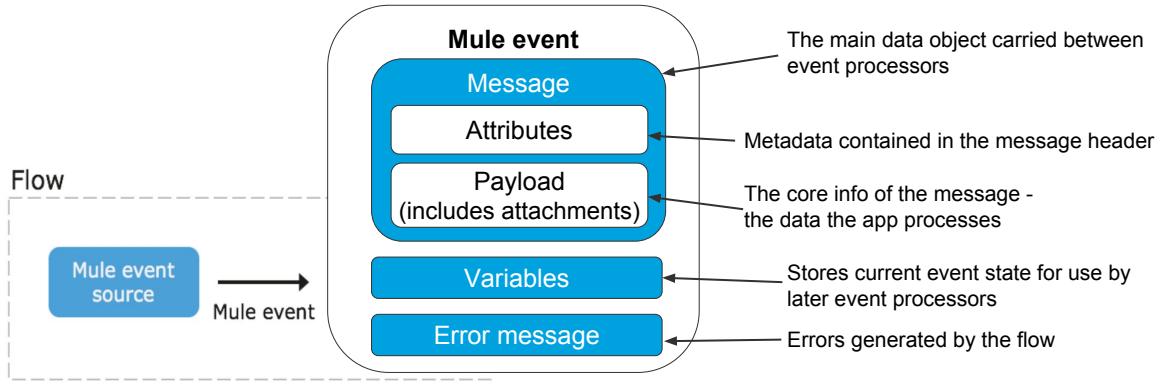
10

Event sources create Mule 4 events at the start of a flow



- A flow can start with an **event source**
 - Accepts inbound events and converts them to **Mule events**
 - The Mule event is passed or copied between event processors in the flow

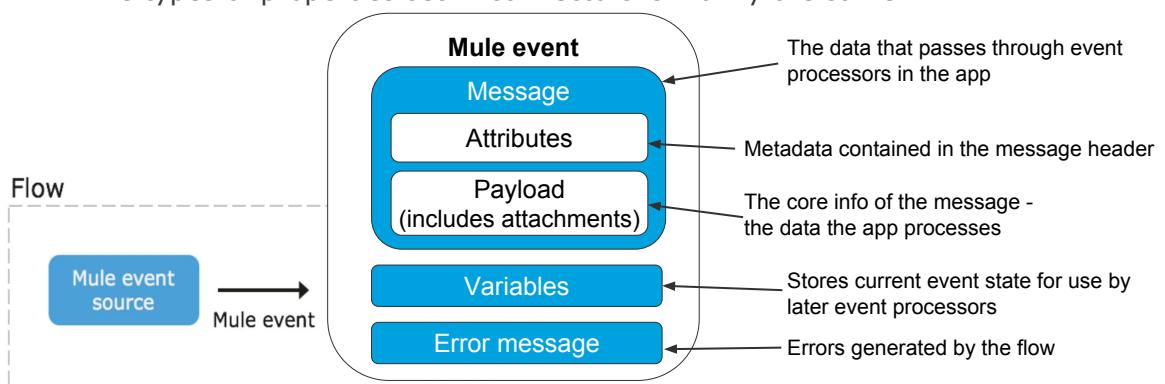
Note: In Mule 3, event sources were called message sources



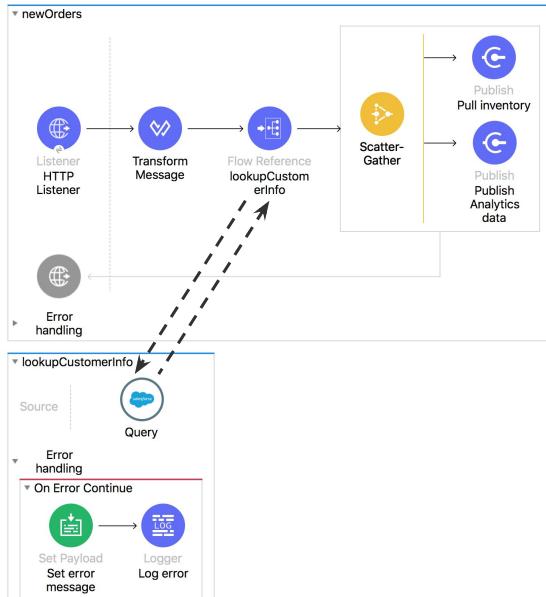
Comparing the structure of Mule 4 attributes vs. Mule 3 properties



- Mule 3 messages had inbound and outbound properties
- In Mule 4, there are only attributes, and each connector is now responsible for setting all outbound properties
 - The types of properties set in connectors is mainly the same



The structure of flows

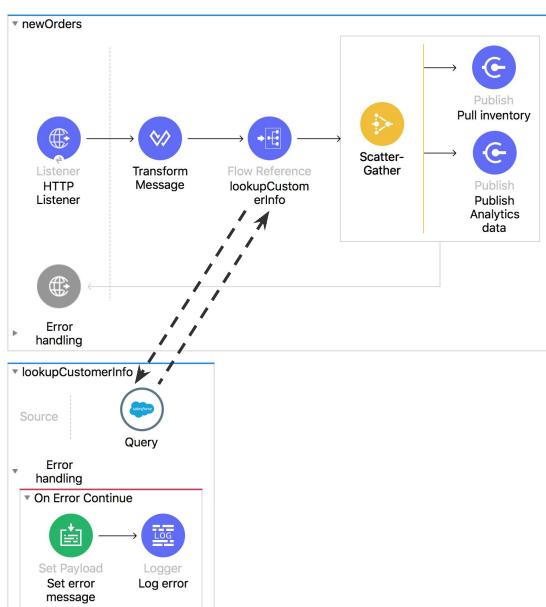


- Each flow

- Can have at most one event source
- Must have at least one event processor
- Can have an error handler
- Can invoke other flows

13

How Mule 4 events progress through event processors in a Mule application

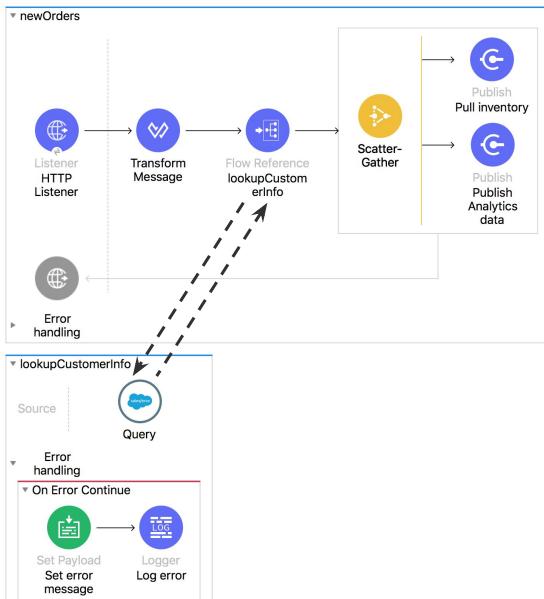


- Flows always function synchronously

- Each event processor always outputs a Mule event, which might be the same as the input Mule event
- Some connector operations (such as **async**) copy the Mule event for concurrent processing (on a new thread)
- To call a sequence of one or more event processors **asynchronously**, for concurrent processing, wrap them in an **async scope**

14

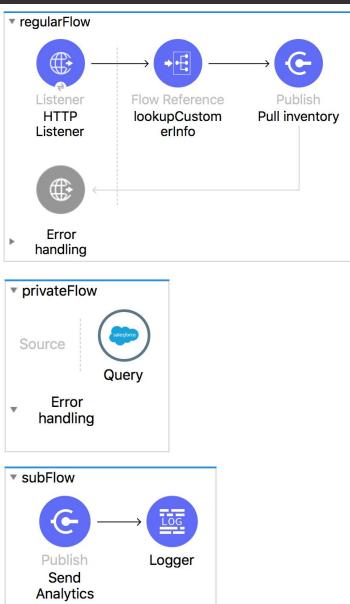
Mule 4 vs. Mule 3 flows



- In Mule 3, flows could define a processing strategy to decide if the flow processing was synchronous or asynchronous
- Flows used various Staged Event Driven Architecture (SEDA) queues
- In Mule 4, asynchronous processing strategies and thread pools are automatically selected by the self-tuning Mule 4 runtime
 - More on this later in the course

15

Three types of flows



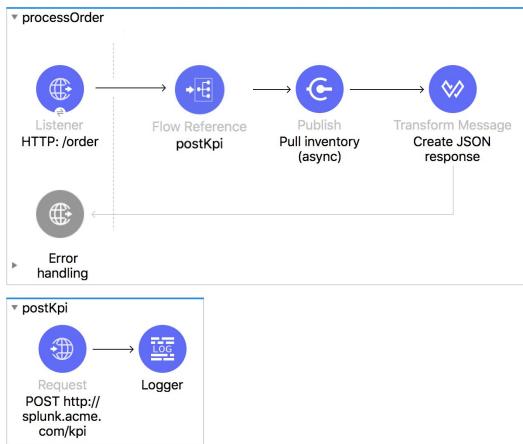
- **Regular flow**
 - Starts with a message source
 - Can have its own error handler
- **Private flow**
 - Identical to a regular flow, but no message source
 - Can only be triggered from within the Mule application
 - Such as through a flow reference or a DataWeave lookup function
- **Sub flows**
 - No message source, like a private flow
 - No error handling
 - Errors bubble up into the parent flow

16

Flows can call other flows



- Flows can synchronously call other flows or subflows via a flow reference
 - In that case, the called flow's message source is skipped synchronously

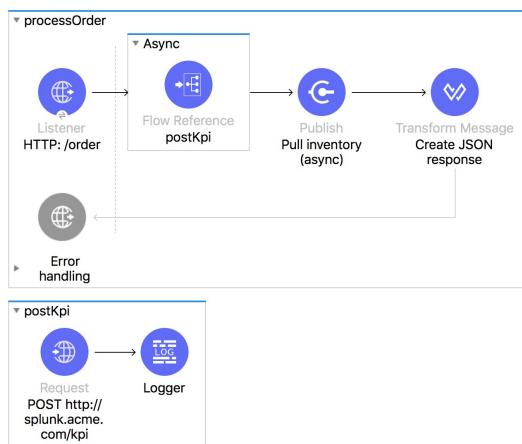


17

Calling flows asynchronously



- All types of flows can be triggered asynchronously from a parent flow using an Async scope



18

Internally triggering a Mule 4 flow using a scheduler



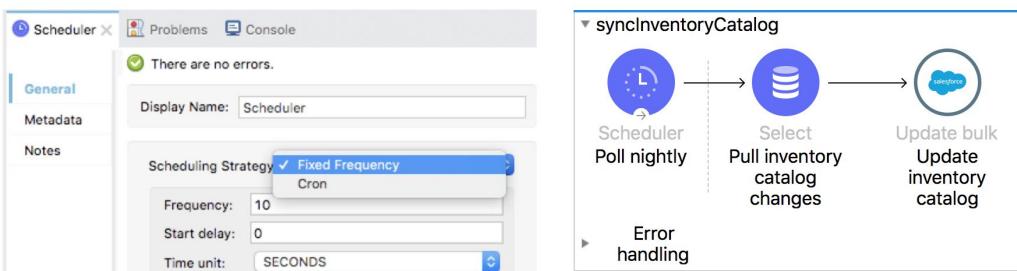
- The **Scheduler** component can trigger a flow at a specific **time**

- Fixed frequency**

- The default is to poll every 1000 milliseconds

- Cron**

- A standard for describing time and date information
 - Can specify an event to occur just once at a certain time or at specific dates or at some frequency

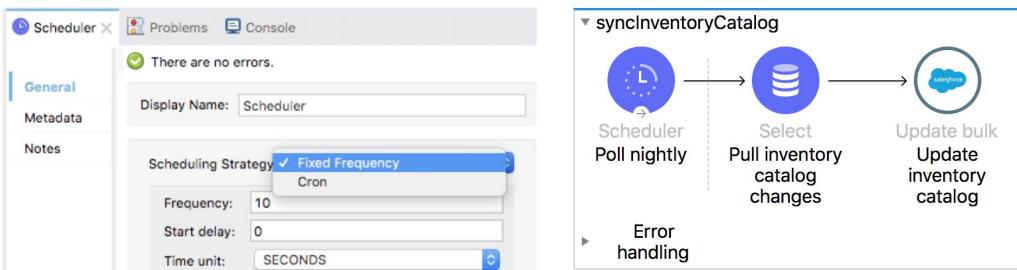


19

Mule 3 uses polling scopes instead of schedulers



- Mule 3 flows could wrap message sources in a polling scope
- The behavior of a scheduler and a polling scope are similar
- In Mule 4, some connectors themselves provide operations to handle the polling and watermarks

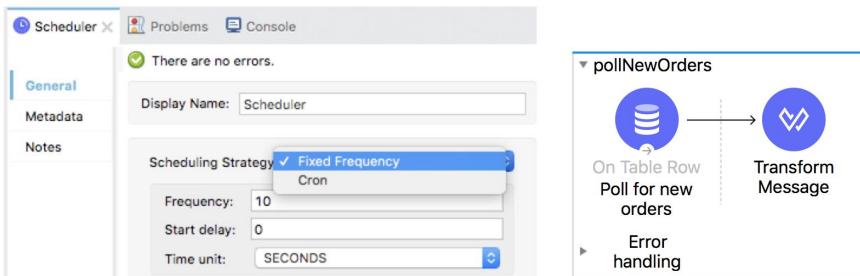


20

Configuring schedulers inside some message sources



- Some connector operations use an internal and implicit scheduling strategy to trigger a flow, such as
 - Database: On Table Row
 - File, FTP, SFTP: On New or Updated File



All contents © MuleSoft Inc.

21

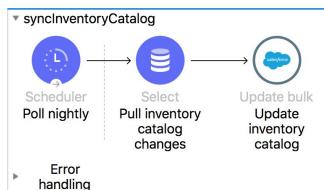
Exchanging data with external resources using Anypoint Connectors



Reviewing Anypoint Connectors



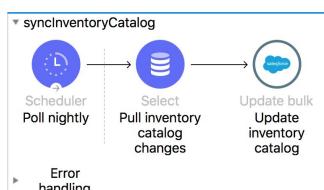
- Anypoint Connectors **abstract and simplify** connections to third party resources and other protocols
 - Built using a common Mule SDK
 - Helps to build new custom connectors in a consistent way that can be shared in Anypoint Exchange
 - For a REST API, an Anypoint connector can be automatically generated in Anypoint Exchange from a RAML file, using the Anypoint REST Connect feature
- In Anypoint Exchange, connectors are divided up into various categories based on support and distribution rights



How to invoke REST or SOAP services



- An HTTP or Web Service Consumer connector can be used
 - But a **custom connector** can explicitly expose the operations of the web service and simplify some operations, such as authorization and authentication
 - A connector automatically generated by Anypoint REST Connect and stored in Anypoint Exchange is also faster and easier to use vs. an HTTP connector



Advantages of using Anypoint Connectors

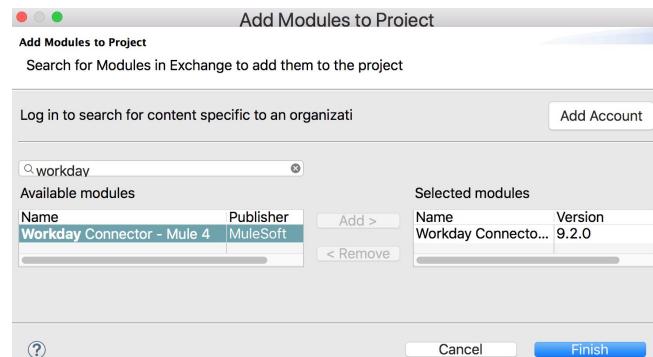
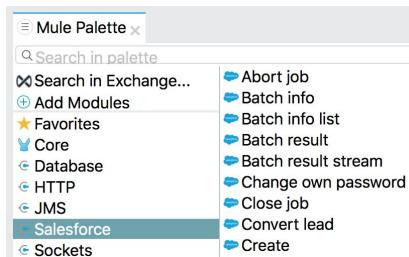


- **Facilitate integration** of Mule applications with **third-party systems** such as Salesforce, Google Apps, Google Cloud, MongoDB, ServiceNow, Workday, Facebook, or Twitter
- No need to study and code to underlying **protocol** or **security** used by third party systems
- **Speed up** development and deployment of Mule applications
- Make mule applications are easier to **Maintain**

Adding Anypoint Connectors to a Mule application



- Anypoint Studio can log in to Anypoint Exchange as a particular Anypoint Platform organization user
 - Search for and install a particular versions of an Anypoint Connector
 - Might include connectors published to the organization's private Anypoint Exchange



- APIkit can auto-generate flows from a RAML file
 - These flows can proxy access to other external services
 - Inside the generated API service flows, external services can be accessed with other Anypoint Connectors
- Once the APIkit flows are implemented, other web clients can connect to the API endpoints using the same RAML specification

Invoking REST APIs using REST Connect

- Other Mule applications can also act as web clients to APIkit-generated flows
 - First upload a RAML or OAS REST API file to Anypoint Exchange
 - Anypoint Exchange then uses its internal Anypoint REST Connect service to auto-generate two complete Anypoint connectors
 - One for Mule 3 and one for Mule 4
 - The connectors are then available for download from the API asset in Anypoint Exchange



Why REST Connect generated connectors are helpful to Mule application developers



- The connector reflects the RESTful API with its resources and methods
 - If invalid parameters or data is input to the connector, specific context sensitive **error messages** are generated
 - Defines **specific metadata** to help with data transformation
 - Such as when using a Transform Message component to visually map DataWeave code
 - This helps Mule applications **fail fast and helps to quickly identify issues** between distributed API endpoints

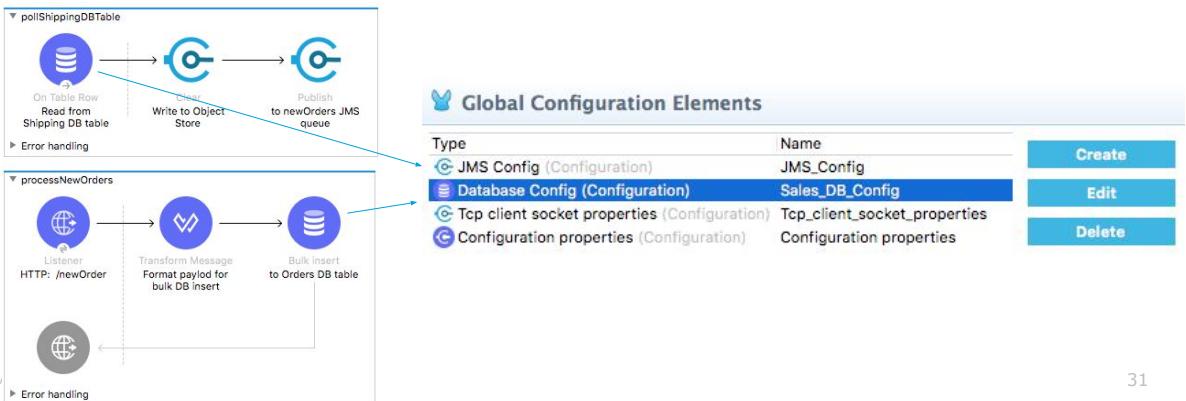
Sharing connections and configurations



Sharing connections and other configurations



- **Global elements** are connectors, caches, and other configurations that can be shared between Mule components
 - Global elements can be **shared** among event processors of the same type
 - For example, a database connection shared by several database operations

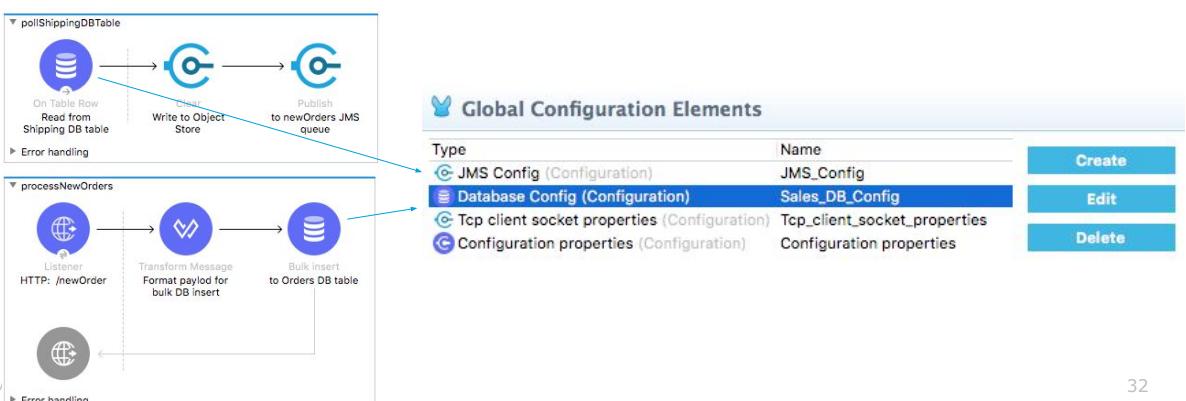


31

Benefits of using global elements



- Global elements simplify the operations and maintenance of the Mule application
 - Change once in the global element, then changes apply to all Mule flow elements that reference the global element

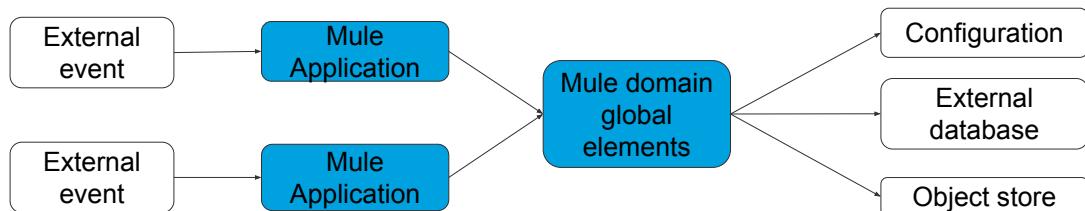


32

Mule domains can share global elements across Mule applications



- Global elements can also be added to a Mule domain project
 - One or more Mule applications can be associated with the same Mule domain
 - A Mule domain can be deployed to Mule runtimes along with Mule applications
 - So now Mule applications deployed to the same Mule domain can now share any connection type or other global element
 - Simpler operations and maintenance
 - Mule domains are only supported in **customer-hosted** runtime planes



All contents © MuleSoft Inc.

33

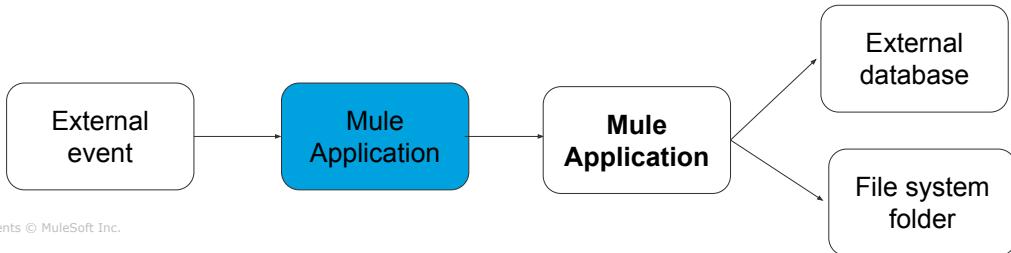
Comparing Mule 4 and Mule 3 applications



What changed in Mule 4?



- Mule 4 and Mule 3 applications are very similar
- Messages are now called events, and the structure changed
- Inbound and outbound properties are called attributes
- Each Mule 4 connector explicitly sets its own outbound properties
- Session/record variables are removed, and instead each connector passes data over transports as Payload, headers, attachments, etc.
- Variables are now stored and passed around with the Mule event



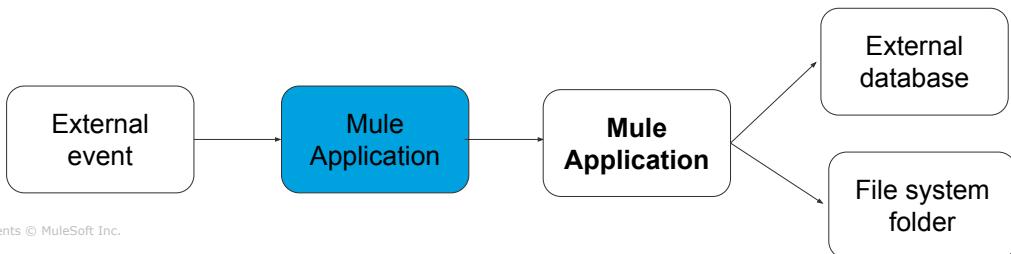
All contents © MuleSoft Inc.

35

How flows changed in Mule 4



- The Mule 4 runtime is self-tuning so SEDA queues are removed
- Flows do not need to set processing strategies



All contents © MuleSoft Inc.

36

Reusing global elements in Mule applications



- Create reusable configuration details for connectors, caches, other components, and other shared configurations
- Can be shared by multiple components in the Mule application or Mule domain
 - Share a global configuration among processors of the same type
- Maintainability
 - Change once in the global element, and the changes apply to all Mule flow elements that reference the global element

The screenshot shows a table titled "Global Mule Configuration Elements". The columns are "Type", "Name", and "Description". To the right of the table are three buttons: "Create" (highlighted with a red box), "Edit", and "Delete". A red circle with the number "2" is positioned above the "Create" button. Below the table, there are navigation links: "All contents © MuleSoft Inc.", "Message Flow", "Global Elements" (highlighted with a red box), and "Configuration XML".

37

Mule 4 vs. Mule 3 connectors



- Unlike Mule 3, Mule 4 applications are **always** Maven projects
- Anypoint Exchange access is integrated into Anypoint Studio
 - New connectors and modules can be added directly into Anypoint Studio from Anypoint Exchange, within a Mule project

The screenshot shows the "Add Modules to Project" dialog. On the left, the "Mule Palette" is open, showing various connectors like "Search in Exchange...", "Favorites", "Core", "Database", "HTTP", "JMS", "Salesforce" (highlighted with a red box), and "Sockets". The main area is titled "Add Modules to Project" with a sub-instruction "Search for Modules in Exchange to add them to the project". It includes a search bar "workday", a "Available modules" list, and a "Selected modules" table. The "Selected modules" table shows one entry: "Workday Connector - Mule 4" by "MuleSoft" with version "9.2.0". At the bottom are "Cancel" and "Finish" buttons.

38

Mule 4 offloads connectors and other components to separate **modules**



- In Mule 3, connectors were bundled with the Mule 3 runtime
 - This created brittle, tightly coupled deployments
 - To upgrade or patch a connector or other component, you had to wait for a new Mule runtime release
 - For customer-hosted Mule runtimes, you also had to download and reinstall the new Mule runtime version
- In Mule 4, each connector and many other components are separated out into independent extension libraries, called **modules**
 - This avoids version conflicts between libraries from different modules

How modules are configured in a Mule application



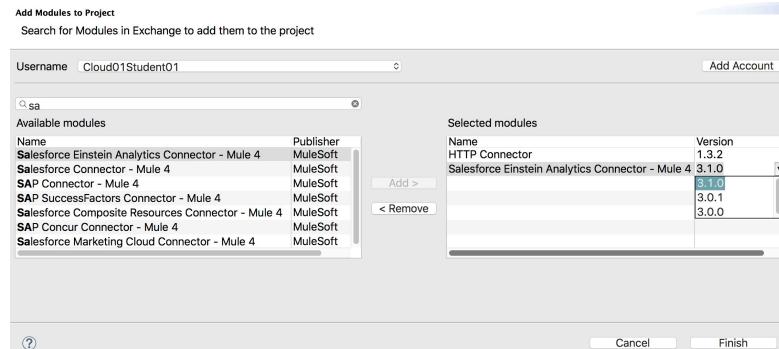
- Each **module is configured as a Maven dependency** in the Mule application's pom.xml file
 - New module versions can be independently downloaded from Maven repositories without needing to update the Mule runtime or any other modules

```
<dependencies>
    <dependency>
        <groupId>org.mule.connectors</groupId>
        <artifactId>mule-http-connector</artifactId>
        <version>1.3.2</version>
        <classifier>mule-plugin</classifier>
    </dependency>
    <dependency>
        <groupId>org.mule.connectors</groupId>
        <artifactId>mule-sockets-connector</artifactId>
        <version>1.1.2</version>
        <classifier>mule-plugin</classifier>
    </dependency>
    <dependency>
        <groupId>org.mule.connectors</groupId>
        <artifactId>mule-objectstore-connector</artifactId>
        <version>1.1.1</version>
        <classifier>mule-plugin</classifier>
    </dependency>
```

The classloading lifecycle of Mule 4 modules



- Each Mule 4 module now has an independent and isolated class loading lifecycle
 - These are supported by a more complex connected graph of classloaders
 - This is the mechanism to support different versions of the same Java libraries (with the same classes and interfaces) in the same Mule application without conflicts



42

Reviewing the behavior of common connectors



- Anypoint Studio ships with some common connectors
- New connectors can be easily downloaded and added from Anypoint Exchange, directly inside Anypoint Studio
- Common connectors include HTTP, Database, File, and FTP

The Mule HTTP connector

- Supports both client and server operations
 - Listen for inbound requests over HTTP/HTTPS to trigger a flow
 - Send an HTTP/HTTPS request in the middle of a flow
 - HTTP 1.0 and 1.1 (HTTP 2.0 is not supported)

The Mule HTTP connector uses the Grizzly libraries internally

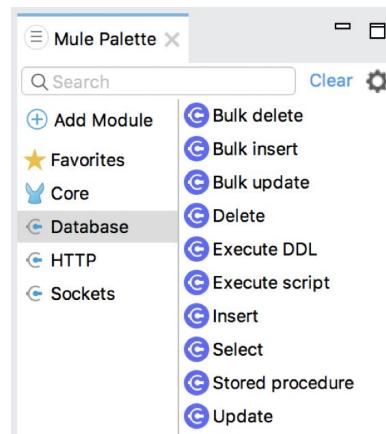


- Grizzly is a standard open source library supporting the HTTP/S protocols
- Uses **selector** thread pools
 - Selector threads check the state of the NIO channels and creates and dispatches events when they arrive
 - Listener selectors poll only for **request events**
 - Requester selectors poll only for **response events**
- HTTP Listener has a **shared selector pool** for all Mule applications deployed to a particular Mule runtime
- HTTP Requester has a **dedicated selector pool** for each Mule application

Database connectors



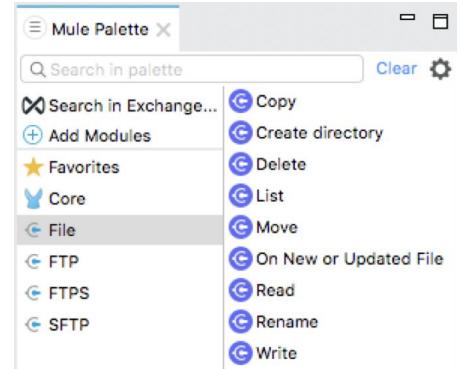
- Can call out to almost any JDBC relational database
 - Any database engine for which you have a JDBC driver
- Support for
 - DDL
 - DML
 - Stored procedure
 - DB script
 - Bulk operation



File and FTP connectors



- These four connectors can work 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
 - Overwriting, appending, and generating new files



All contents © MuleSoft Inc.

48

SaaS connectors



- There are many dedicated connectors such as for SaaS systems
 - Available from Anypoint Exchange
 - Import the connector into Anypoint Studio or flow designer and start integrating



49

SaaS connectors often support many specialized and optimized operations



The figure consists of three side-by-side screenshots of the Mule Palette interface, each showing a list of operations for a specific SaaS connector.

- Mule Palette X (Search: sales cre):** Shows operations like Create, Create batch, Create batch for query, Create batch stream, Create bulk, Create bulk for query, Create job, Create job bulk v 2, Create metadata, Create multiple, Create multiple batch, Create multiple batch stream, Create multiple bulk, and Create single.
- Mule Palette X (Search: sales bul):** Shows operations like Bulk info, Bulk info list, Bulk result, Bulk result stream, Create bulk, Create bulk for query, Create job bulk v 2, Create multiple bulk, Get bulk job state bulk v 2, Hard delete bulk, Retrieve record failure bulk v 2, Retrieve record success bulk v 2, Update bulk, Update multiple bulk, Upsert bulk, and Upsert multiple bulk.
- Mule Palette X (Search: sales dell):** Shows operations like Delete, Delete job v 2, Delete metadata, Delete multiple, Get deleted, Get deleted range, Hard delete bulk, Hard delete multiple, and On Deleted Object.

All contents © MuleSoft Inc.

50

Building custom connectors with the Mule 4 SDK



- The Mule SDK for Java supports building **custom** connectors for Mule 4
 - Custom connectors are used to integrate existing business logic, algorithms, etc.
- You should first prefer OOTB components or web services (REST or SOAP) over custom components
- Build custom connectors only when an API or OOTB connector does **not** support the use case
 - First, look in Anypoint Exchange for a specialized pre-built connector (for example SFDC)
 - If you don't find one, see if you can connect to the external system with a web service (REST or SOAP)
 - Finally, evaluate if it is worth creating a custom connector

All contents © MuleSoft Inc.

51

Migrating Mule 3 DevKit custom connectors to Mule 4



- The Anypoint Connector DevKit is not compatible with Mule 4
- A DevKit migration tool will support migration of custom Mule 3 connectors build with the Anypoint Connector DevKit
 - Will typically also involve some custom coding

Types of Anypoint Connectors and levels of support



Connector type	Description
Community https://anypoint.mulesoft.com/exchange/?search=community	<ul style="list-style-type: none">• MuleSoft or members of the MuleSoft community write and maintain the community connectors• No license
MuleSoft Certified https://www.anypoint.mulesoft.com/exchange/?search=mulesoft-certified	<ul style="list-style-type: none">• Developed by MuleSoft's partners and developer community and are reviewed and certified by MuleSoft• Contact the MuleSoft partner that created the MuleSoft Certified connector for support
Select https://anypoint.mulesoft.com/exchange/?search=select	<ul style="list-style-type: none">• MuleSoft maintains Select Connectors• Connectors included in the open source Mule distribution can be used by everyone, however support is only included in an Anypoint Platform subscription• To use all other Select Connectors and access support, you must have an active Anypoint Platform subscription.
Premium https://anypoint.mulesoft.com/exchange/?search=premium	<ul style="list-style-type: none">• MuleSoft maintains Premium Connectors• You must have an active CloudHub Premium plan or an Enterprise subscription with an entitlement for the specific connector you wish to use.

Configuring reconnection strategies for connectors



Many connectors can configure a reconnection strategy



- Connectivity tests run when the Mule application starts, then periodically while the Mule application runs
- The reconnection strategy dictates what to do when connectivity fails

Connection

Reconnection strategy Standard None Forever

Frequency (ms): 2000

Reconnection Attempts: 2

Mule Palette X

Search Clear

Add Module Favorites Core Database HTTP Sockets

Bulk delete Bulk insert Bulk update Delete Execute DDL Execute script Insert Select Stored procedure Update

- By default, a failed connectivity test is just logged and the Mule application **starts anyway** or **continues to run** without **trying to reconnect**
- However, a **reconnection strategy** can be configured on some connector operations to instead repeatedly try to connect
- A **failsDeploy** attribute can be set to **true** to **throw an exception** if the reconnection attempts fail, which **prevents the Mule application from starting**

```
<http:request-config name="HTTP_Request_Config" doc:name="HTTP Request Config">
    <http:request-connection host="https://jsonplaceholder.typicode.com/posts" port="80" >
        <reconnection failsDeployment="true">
            <reconnect frequency="4000" count="5" />
        </reconnection>
    </http:request-connection>
</http:request-config>
```

All contents © MuleSoft Inc.

56

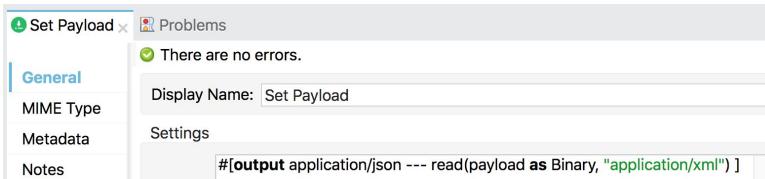
Transforming data using DataWeave



Transform data structures using DataWeave 2



- DataWeave is a functional programming language to convert input data to new and restructured output data
- DataWeave 2 is the main **expression language** used everywhere in Mule 4 to access, query, and transform data
 - Mule 3 data format transformers like object-to-string are no longer needed
 - In Mule 4, data formats can be converted directly in-place in the event processor attribute's value



All contents © MuleSoft Inc.

58

DataWeave is an any-to-any transformation language



- All input data types (such as XML, JSON, Java, CSV files) are represented by a common data-type-neutral data model
- All DataWeave expressions are applied to and execute against the DataWeave representation of input data

The **header** contains directives that apply to the body expression

Input	Transform	Output
{ "firstname": "Max", "lastname": "Mule" }	%dw 2.0 output application/xml --- { user: { fname: payload.firstname, lName: payload.lastname } }	<?xml version="1.0" encoding="UTF-8"?> <user> <fname>Max</fname> <lname>Mule</lname> </user>

Delimiter to separate header and body

The **body** contains a DataWeave expression that generates the output structure

All contents © MuleSoft Inc.

59

DataWeave expressions always evaluate to a DataWeave type



- DataWeave expressions always evaluate to a valid DataWeave type
 - **Objects:** Represented as collection of key-value pairs
 - **Arrays:** Represented as a sequence of comma separated values
 - **Simple literals:** String, Number, Date, Time, DateTime, Boolean
- The output type of a DataWeave expression is declared in a header
 - Such as XML, JSON, Java, CSV

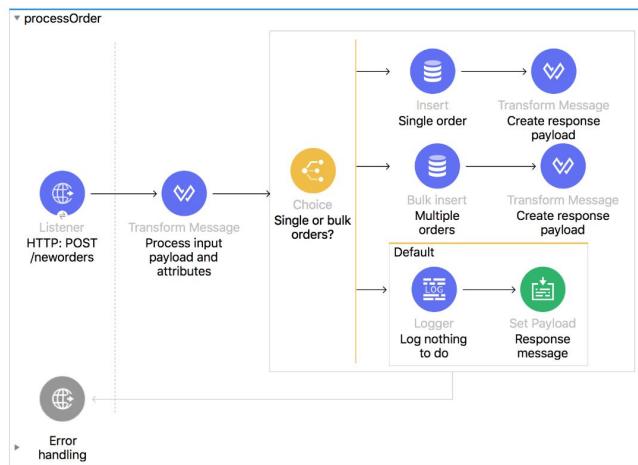
Routing Mule events between event processors



Controlling and routing Mule events in flows using routers



- A router uses particular logic to send events to one or more **sequences of event processors** (also called **routes**)



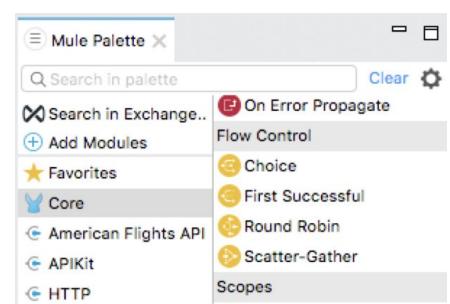
All contents © MuleSoft Inc.

62

Types of Mule routers



- Choice
 - One route is executed based on conditional logic
 - The logic is a DataWeave expression usually based on the inbound Mule event
- Scatter-Gather
 - All routes are executed in parallel with copies of the same inbound Mule event
- First Successful
 - Each route is executed sequentially until one is successfully executed
- Round Robin
 - One route is executed each time
 - The next route is selected by iterating through a list maintained across executions
- Error handling
 - Re-routes event processing to try to handle various errors



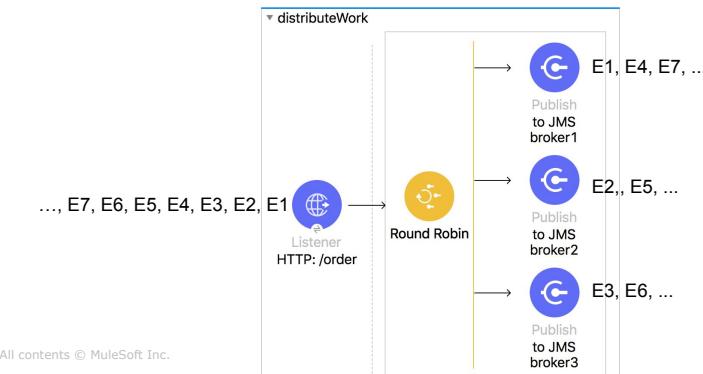
All contents © MuleSoft Inc.

63

Round-robin Mule event processing



- Routes the incoming Mule event to one of its routes
- A different route is selected each time in a looping round-robin manner
- Each invocation of the Round Robin router is synchronous



All contents © MuleSoft Inc.

64

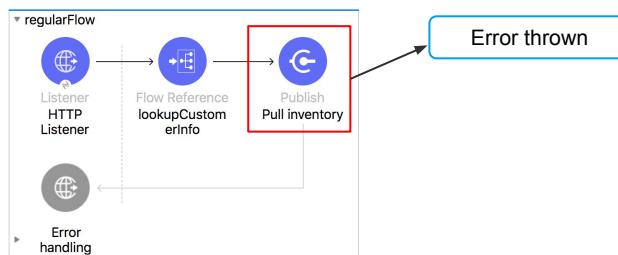
Reviewing how errors are generated by flows



How errors are generated in flows



- Each event processor might throw an error
- The error might originate from a Java library in the event processor's source code, or from a DataWeave expression, or other source
- If an error is not handled, the error message is logged and the flow processing stops
 - If the flow has an event source, a response error message is usually returned



All contents © MuleSoft Inc.

66

Mapping error types in Mule components



- Many Mule components allow mapping errors to a new type
- The new type is a custom namespace and error type
 - Allows the error to abstract and apply context to underlying failures
 - Cannot use an existing error namespace from another Mule component in the Mule application
 - For example, cannot map FILE:FILE_NOT_FOUND to HTTP:UNAUTHORIZED

A screenshot of the Mule Studio interface. The top bar shows tabs for 'to JMS broker2', 'Problems', 'Console', and 'Mule Debugger'. The 'Error Mapping' tab is selected in the sidebar, which also includes 'General', 'Advanced', 'Metadata', and 'Notes' tabs. The main area displays a message: 'There are no errors.' Below this is a table with two columns: 'Error types to be mapped:' and 'Map to:'. The 'Error types to be mapped:' column contains 'JMS:PUBLISHING, JMS:SECURITY'. The 'Map to:' column contains 'MYAPP:CONNECTION_ERROR'. There are '+' and '-' buttons next to the mapping table.

All contents © MuleSoft Inc.

67

- The Raise Error component can throw a standard or custom error type

```
<raise-error type="ACCOUNT:INSUFFICIENT_FUNDS"  
description="#['Cannot transfer $(payload.amount) since only  
$(vars.balance) are available.']"/>
```
- The description is the message in the thrown error
- The error type must be
 - A core MULE error type (with no namespace)
 - Example: SECURITY (note MULE:SECURITY)
 - A custom namespace
 - Example: MYAPP:BAD_DATE

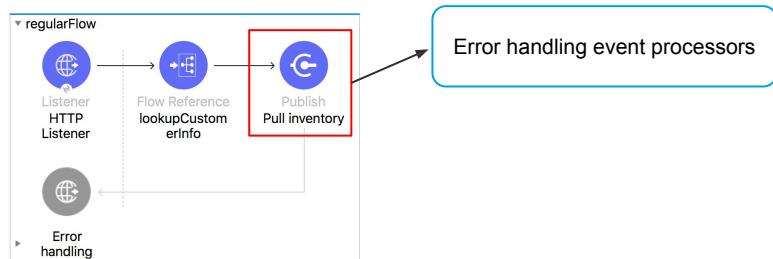
Reviewing Mule application error handling



How errors generated in a Mule 4 flow are handled



- When an event is being processed through a Mule flow that throws an error
 - Normal flow **execution stops**
 - The Mule event is passed to the first processor in an **error handler**
 - Depending on the type of error handler(s) configured, the error is either caught and handled, or is thrown up the flow invocation chain



All contents © MuleSoft Inc.

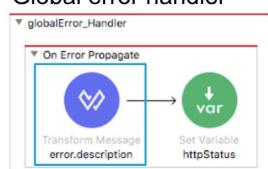
70

Where an error handler can be added in a Mule application

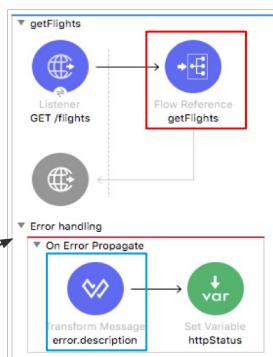


- An error handler can only be added to
 - A global error handler (outside of any flows)
 - A flow
 - A Try scope
 - A sequence of one or more Mule events
- Not to subflows

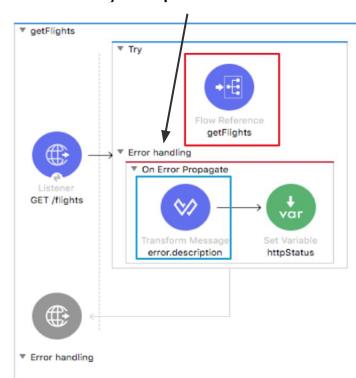
Global error handler



Flow error handler



Try scope error handler



All contents © MuleSoft Inc.

71

How default error handling works

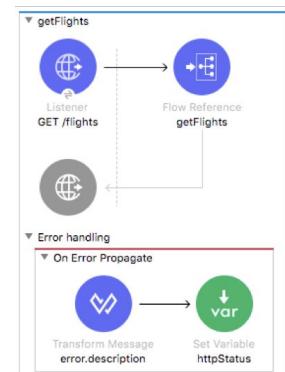
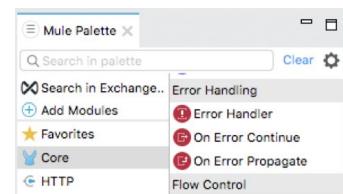


- If there is no error handler defined for a flow or try scope, a **Mule default error handler** is used
- The Mule application can configure a global error handler as its default error handler
- Otherwise the Mule runtime provides its own default behavior
 - Implicitly and globally handles all messaging errors thrown in Mule applications
 - Stops execution of the flow and logs information about the error
 - Automatically rethrows the error
 - Cannot be configured

Error handling



- Each error handler can contain one or more error scopes
 - On Error Continue scope
 - On Error Propagate scope
- Each error scope can contain any number of event processors



Comparing how the two types of error scopes behave



- **On Error Propagate scope**
 - All processors in the error handling scope are executed
 - At the end of the scope
 - The rest of the flow that threw the error is not executed
 - *The error is re-thrown up to the next level and handled there*
 - If the erroring flow starts with an HTTP Listener, it returns an **error** (5xx) response
- **On Error Continue scope**
 - All processors in the error handling scope are executed
 - At the end of the scope
 - The rest of the flow that threw the error is not executed
 - *The result of the error handler scope is passed up to the next level as if the flow execution had completed successfully*
 - If the erroring flow starts with an HTTP Listener, it returns a **success** (2xx) response

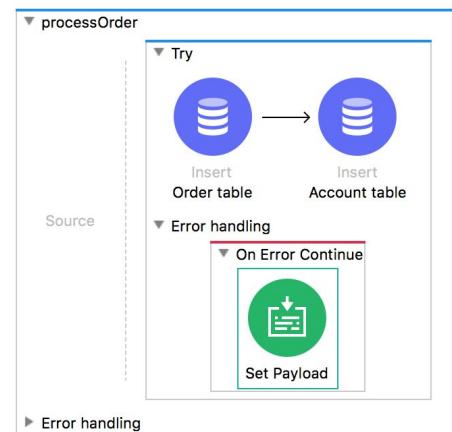
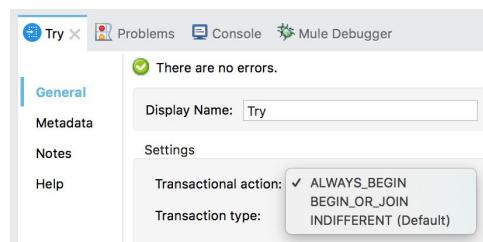
All contents © MuleSoft Inc.

74

Error handling within a transaction



- On Error Propagate scope
 - The error is thrown up the call stack (propagated), so the current open transaction is rolled back
- On Error Continue scope
 - The error is handled, so the current open transaction is committed



All contents © MuleSoft Inc.

75

Setting global error handlers for Mule applications



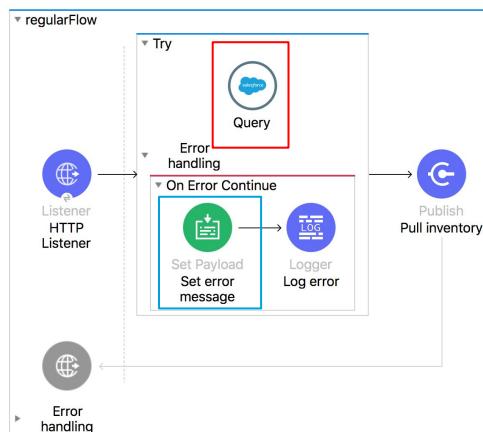
- A global error handler can be configured in a Configuration global element for the entire Mule application
 - This error handler is then used by any flow or Try scope that does not configure an error handler
- A Configuration global element can also be added to a Mule domain
 - Then every Mule application in the Mule domain uses the same global error handler instance

The screenshot shows the 'Global Mule Configuration Elements' interface. On the left, there's a navigation bar with 'Message Flow', 'Global Elements' (which is highlighted with a red box), and 'Configuration XML'. Below the navigation is a table with columns 'Type', 'Name', and 'Description'. A 'Create' button is highlighted with a red box and has a red circle with the number '2' above it. To the right is the 'Global Element Properties' panel, which includes tabs for 'General', 'Notes', and 'Help'. Under the 'General' tab, there's a 'Settings' section with a 'Default Error Handler' dropdown set to 'myGlobalErrorHandler'. At the bottom right of the interface is the number '76'.

Grouping sequences of event processors with a **try scope**



- Acts like a private flow inside the parent flow
 - Can have its own error handling
 - Unhandled errors bubble up (out) to the parent flow



- When an error does not involve a Mule event, the system error handler is invoked
 - Examples
 - The Mule runtime has errors while starting up, like running out of memory
 - If the connection for a Database connector goes offline or the network disconnects
- The system error handler
 - Logs the error
 - If the error was caused by a connection failure, the connector's reconnection strategy is executed
- These system errors are **not** caught by any flow or Try scope's error scopes, nor by any configured global error handlers

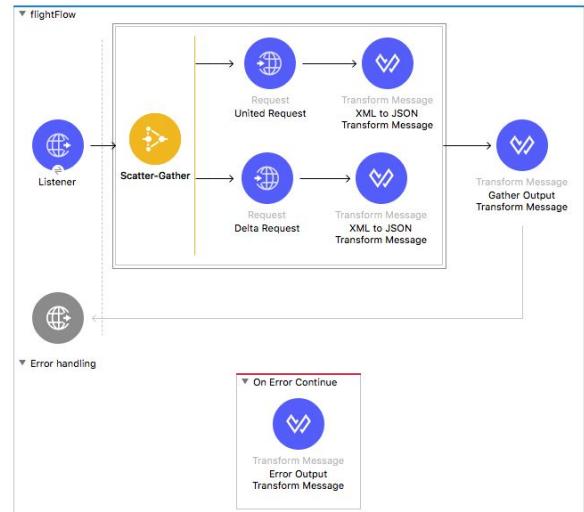
Applying Mule application components to the course case study



Design Mule applications to consume and process several APIs



- You have now seen the basic building blocks of common Mule applications
 - Connectors
 - DataWeave transformations
 - Flow control
 - Error handlers
- Now you will design solutions for common use cases
- Start with designing how to consume and process two APIs



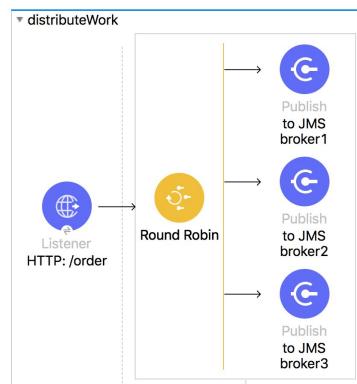
All contents © MuleSoft Inc.

80

Exercise 3-1: Design an integration application that consumes and combines two external APIs



- Design Mule flows to retrieve data from two separate APIs
- Design Mule flows to combine two API XML responses together
- Design Mule flows to translate an XML API response to JSON



All contents © MuleSoft Inc.

81

- Getaways needs to integrate flight information into their travel bundles
- This project is similar to the Development Fundamentals use case
- In this case though, the United and Delta airline APIs expose flight information as a REST service with XML formatted responses

- You need to design an integration solution that fulfills these functional requirements
 - Call the United and Delta airline REST APIs
 - Process each flight API's result information
 - Combine the airline API XML responses together
 - Return the combined response as JSON
- For this exercise, do not worry about particular SLAs relating to latency or data throughput
 - You will address SLAs and other NFRs later in the course

Exercise steps required to build the solution



- Identify flows to fulfill all the integration requirements
- Identify Mule components for each flow, and their interactions across flows
- Identify how to transform and combine data
- Identify error handling for the solution
- Confirm your design meets the integration requirements
 - Call the United and Delta airline REST APIs
 - Retrieve flights information from each airline's API
 - Combine the airline API XML responses together
 - Return the combined response as JSON

Exercise steps: Design the initial data gathering stages



- Identify components to use in the integration flows
 - How should the entire integration process be kicked off, and what are the tradeoffs of each option?
 - What components should retrieve data from the two REST APIs?
 - What components can be used to orchestrate the integration of the responses from the REST APIs?
 - Implement a POC of the integration solution using flow designer or Anypoint Studio
 - Do not worry about connecting to any real data source

Exercise steps: Design the initial data gathering requirements



- Integration requirements
 - Call the United and Delta airline REST APIs
 - Retrieve flights information from each airline's API
 - Combine the airline API XML responses together
 - Return the combined response as JSON

Exercise steps: Design the data aggregation and post-processing stage



- How does the data from each REST API need to be transformed, if at all?
- What are options for combining the flights data together, and what are the tradeoffs of these options?
- What Mule components or custom code are needed for these options?
- Pick the most idiomatic (used for its intended purposes) options for this use case
- What are the best options to transform the XML responses to JSON

Exercise steps



- Decide on error handling options
- Analyze the relative merits of each error handling option
 - Add error handling to your POC flows
- Analyze how you could reuse error handling logic between flows
- Document the tradeoffs of different types of error scopes in your POC
- If an external client triggers the integration process, how are error statuses returned to the client?

Exercise steps



- Identify scopes to use in the integration flows
 - Identify interdependencies between components in the Mule application
 - Can we process independent components in a different construct?
 - Analyze the impact of these optimizations

Exercise worksheet



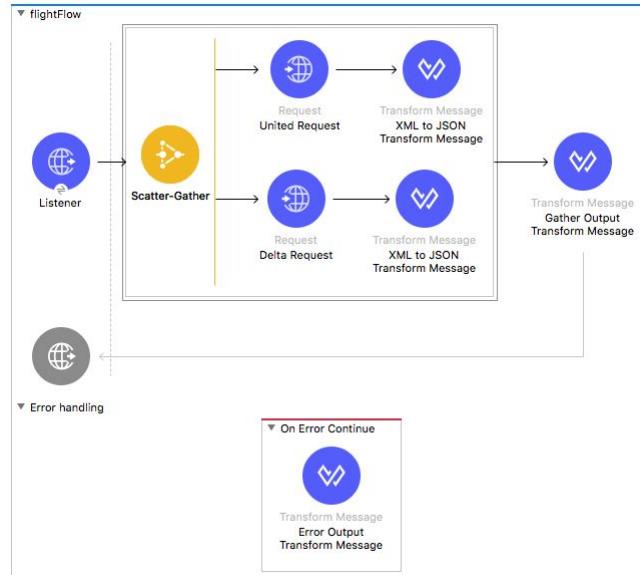
Question	Current solution	Ways to improve
How does the data from each REST API need to be transformed, if at all?		
What are options for combining the flights data together, and what are the tradeoffs of these options?		
What Mule components or custom code are needed for these options?		
Pick the most idiomatic (used for its intended purposes) options for this use case What are the best options to transform the XML responses to JSON		

Exercise worksheet



Related components	Independent components	Ways to improve the design	Impact of the design change

Exercise solution



Exercise solution

- The flight request is split through a Scatter-Gather
 - The United and Delta web services are called in parallel
 - Each response is transformed to a common Flights JSON schema
 - After both transformed web service responses complete, the result is automatically gathered into a single Mule event
- After the Scatter-Gather, the gathered responses are further transformed to flatten each flight result into one common object
- Errors are handled inside the flow
- The On Error Continue allows this flow to handle its own errors
 - This will hide the error if this flow is called from a parent flow

Exercise reflection: Analyze your solution choices against other options



- Did you use a generic HTTP or REST connector, or is there a more targeted connector available in Anypoint Exchange?
 - Document your reason for your connector choices
- Did you use a Scatter-Gather component to perform the two API requests in parallel?
 - Document your reason for this
 - If you used a Scatter-Gather component, how does this affect the error handling?
- How are the two API responses transformed to JSON?
 - Did you pre-process each API response first, then combine to JSON, or did you transform them all at once in a single transformation?
 - Document the tradeoffs for your data transformation choices

Exercise reflection: Analyze error handling choices and tradeoffs



- How did you handle errors?
 - Document the tradeoffs to using a common global error handler vs. more localized error handling (such as with Try scopes)
- Did you use any error mappings?
 - Document your reasons to use or not to use error mappings
 - Document your reasons for your choices of return statuses

(Optional) Exercise 3-2: Design an application that one-time-loads data into a database from files



- Design a Mule application to load data from a file into a database.

Exercise 3-3: Design an integration application that synchronizes a database and SaaS system



- Design a Mule application to load data from a file into a database
- Design a Mule application to poll a database for changes
- Design a Mule application to only transform new database entries
- Design Mule flows to translate a database response to Salesforce operations
- Design Mule flows to call Salesforce operations in bulk

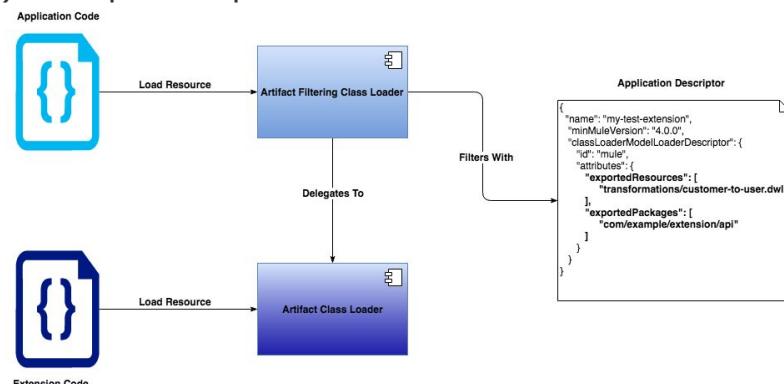
- Continue to design Mule applications to meet the course case study
- Perform a similar analysis as Exercise 3-1 for the DB to SFDC synchronization use case
 - Identify where Mule application components can help integrate between the database and the Salesforce systems
 - Identify which connectors can be used
 - Identify where schedulers or batch jobs might be applied
 - Identify the types of data transformation that may required, and how DataWeave can help
 - Identify how errors are handled

How Mule 4 isolates class loading



Mule 4 class loading isolation

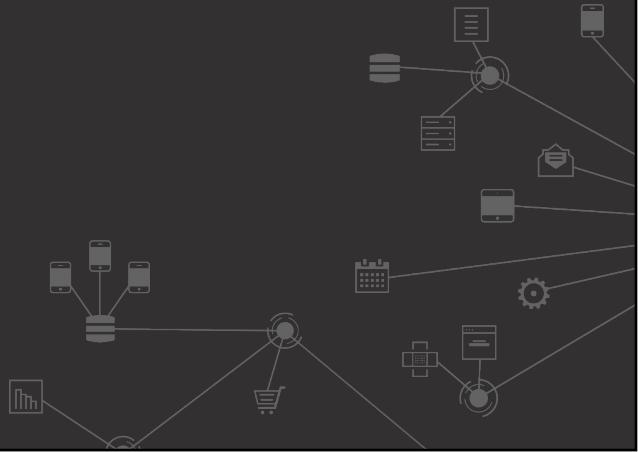
- Mule 4 uses separate class loaders to isolate the Mule runtime, Mule applications, and modules from each other
- Each class loader specifies the packages (rather than classes and resources) to export as part of the module's interface



Class loading isolation - Mule 4

- **Artifact class loader**
 - A regular Java class loader pointing to the JAR files included in the module
 - This class loader will load all files and classes of the module
- **Artifact filtering class loader**
 - A wrapper created over the Artifact class loader, which will enforce the access restrictions to the module's code for foreign artifacts (the app or other plugins)
 - It uses the content of the mule-artifact.json descriptor to determine what is public
- **Module code**
 - Plugin, modules and connectors
 - Uses Artifact class loader (which does not have any restriction), and it is only able to locate resources of the plugin itself
- **Application code**
 - Mule app
 - Uses the Artifact Filtering class loader of the module to prevent the app from accessing restricted code or resources.

Summary



Summary

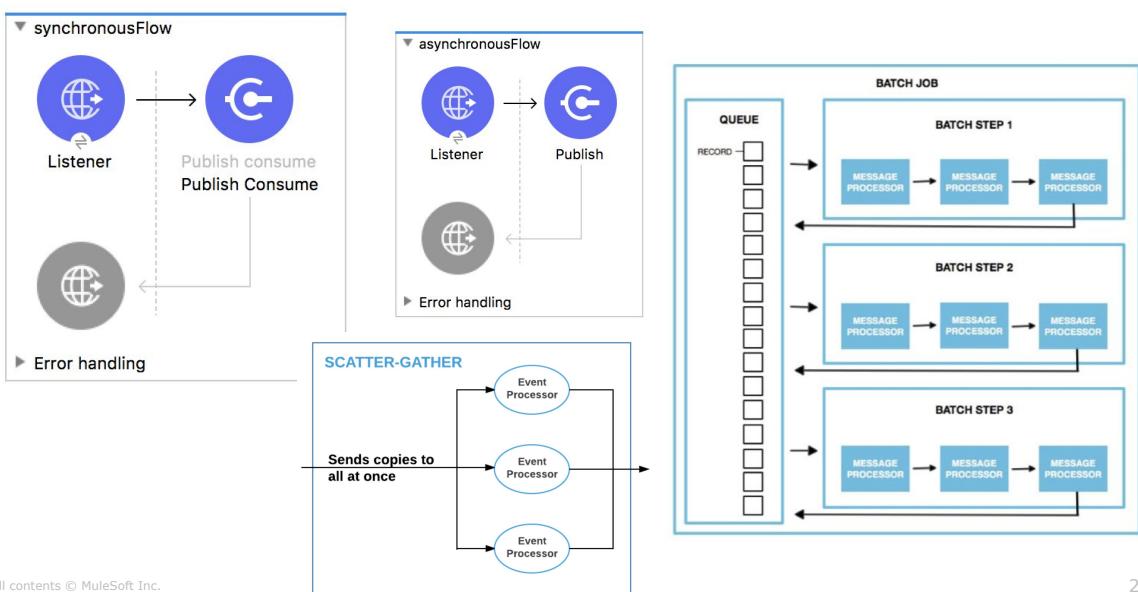


- Mule applications are built from various event processors
- Mule events include the message and variables
- The event's message includes a payload (which includes attachments) and attributes
- Event processors include connectors and Transform Message components
- Event processors can be grouped into various scopes for flow control, async processing, batch processing
- Flows can configure various hierarchies of event handlers

Module 4: Choosing Appropriate Mule 4 Event Processing Models



Goal



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



- Identify and distinguish between Mule 4 event processing models
- Identify what event processing models are used in various Mule 4 scopes and components
- Identify Mule 4 streaming options and behaviors
- Select appropriate processing models for a particular use case

Introducing Mule 4 event processing models and options



Abstracting event processing options into processing models



- **Processing models** collect together options and behaviors related to Mule event processing by Mule application components and flows within a Mule runtime
- Event processing by **Mule scopes** can be modelled to describe
 - How Mule components can process a copy of the same Mule event in parallel, then combine the results
 - How Mule components process a Mule event that contains a collection of records, sequentially or in parallel
- Event processing by **Mule connectors** can be modelled to describe
 - The behavior of inbound Mule event sources at the beginning of a flow
 - How message queuing connectors process Mule events

Modelling Mule 4 event processing



- These models describe the **event processing** behavior of
 - **Non-blocking** and **concurrent** Mule event processing by the Mule 4 runtime
 - **Synchronous** Mule event processing by Mule 4 components and flows
 - **Asynchronous** Mule event processing by Mule 4 components and flows
 - **Parallel processing** of Mule events by Scatter-Gather components
 - **Streaming processing** of larger-than-memory Mule events
 - **Iterative processing** of Mule events containing collections of records
 - **Real-time** and **scheduled** event processing of Mule events

Describing non-blocking and reactive event processing by Mule 4 runtimes



What is **reactive programming**?



- A declarative programming paradigm that combines **concurrency** with **event-based** and **asynchronous** systems
- Popular with web-based distributed systems

References:

https://en.wikipedia.org/wiki/Reactive_programming

<https://www.reactivemanifesto.org/>

Features of reactive programming and reactive streaming



- Uses the best ideas from the Observer pattern, the Iterator pattern, and functional programming
- Deals with an ongoing and building **event stream**
 - Rather than one complete, static, all in-memory data collection
- Is an **asynchronous, non-blocking, and declarative** programming style
 - Avoids the "callback hell" brought about by Java's imperative programming approach
- Incorporates handling of **back-pressure**
 - Automatically slows down Mule event producers if event consumers are being overwhelmed by the rate of events

The Mule 4 runtime uses a non-blocking and reactive processing model



- **Non blocking** is a central theme in Reactive principles
- Non blocking is the norm in Mule 4
 - Every flow has top level support for **non-blocking** operations
 - Although many connector operations are blocking, the Mule 4 runtime makes it easy to develop non-blocking operations
 - For example, the HTTP connector is non-blocking
 - **Threads do not block** waiting for IO intensive operations to complete
 - Unlike Mule 3, the developer does **not** need to assign a **processing strategy** to each flow

The Mule 4 reactive streams use common global thread pools



- In Mule 3, each flow had its own thread pools, SEDA queues, etc.
- Flows had to specify a processing strategy
- In Mule 4, there are three global executors (schedulers) that run all tasks in the Mule runtime
- The Mule runtime's attempts to automatically infer the execution type of each event processor
 - BLOCKING - Operation requires a connection and is blocking
 - CPU_INTENSIVE - Operation requires a connection and is non-blocking
 - CPU_LITE - Otherwise
- The developer of a connector can also explicitly annotate the execution type

The three possible execution types that can be set in an Anypoint connector's source code



- These are set as annotations in the Java classes used to build connectors using the Mule SDK
 - IO_INTENSIVE (BLOCKING)
 - Blocking processing that performs blocking IO operations, all blocking **IO-intensive connectors (e.g. DB via JDBC), Transaction scopes, Lock.lock(),** or any other technique that blocks the current thread during processing without exercising the CPU
 - CPU_INTENSIVE
 - Intensive processing, such as complex **time-consuming calculations, scripting, or DataWeave transformations**
 - This processing type is never automatically inferred by the Mule runtime
 - CPU_LITE
 - Processing that neither blocks nor is CPU intensive, such as **message passing, filtering, routing,** or non-blocking IO

Advantages of the non-blocking Mule 4 runtime



- The Mule 4 runtime leverages **non-blocking IO calls** to avoid performance problems
 - Removes complex tuning of thread pools and their sizes requirement to achieve optimum performance

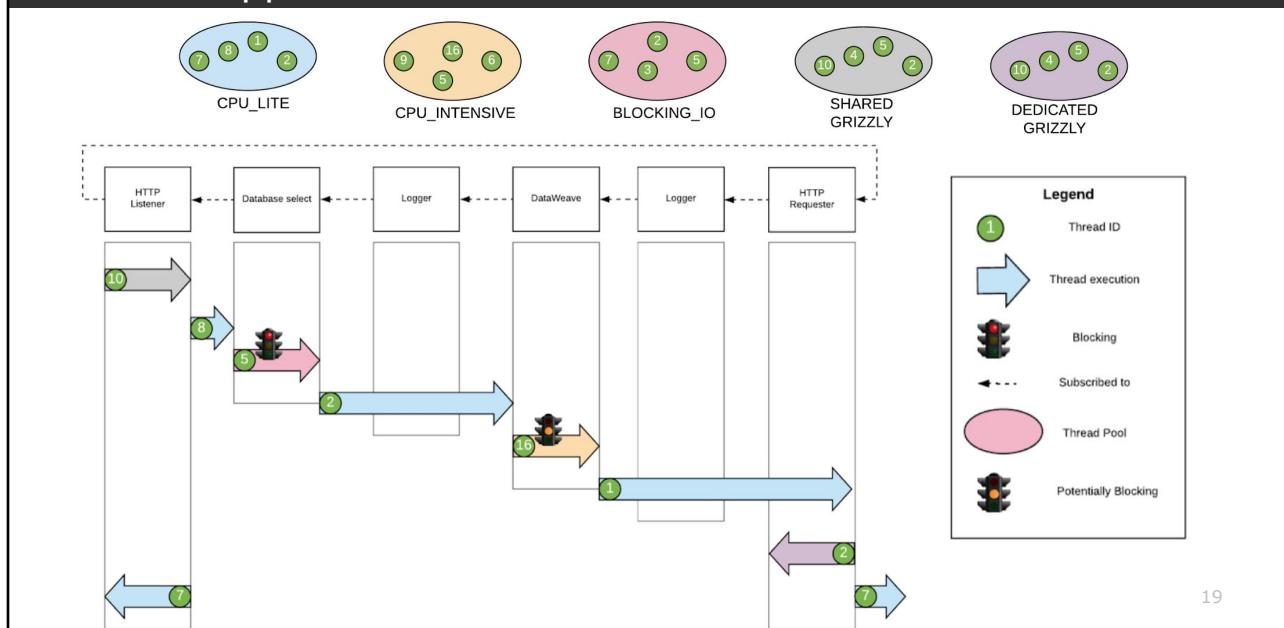
How the Mule 4 runtime self tunes Mule application performance



- The Mule 4 runtime uses a **scheduler execution** service to automatically optimize performance
 - The Mule 4 runtime provides three types of thread pools for use by the event processors in a Mule application's flows
 - The scheduler service assigns each event processor to one of these three thread pools

Thread switching between event processors in a Mule 4 application

MuleSoft



19

Example: Introducing the HTTP connector processing model

MuleSoft

- Mule HTTP connectors use Grizzly under the covers
- Grizzly uses **selector** thread pools
 - Selector threads check the state of the non-blocking IO (NIO) channels and creates and dispatches events when they arrive
 - HTTP Listener selectors poll for **request events only**
 - HTTP Requester selectors poll for **response events only**
- The **HTTP Listener** uses a **shared selector pool** provided by the Mule runtime for **all** Mule applications
- The **HTTP Requester** has a **dedicated selector pool** local to **the** Mule application

- Give some examples of components you feel should be CPU_INTENSIVE, and why?
- Give some examples of components you feel should be CPU_LITE, and why?
- Give some examples of components you feel should be IO_INTENSIVE (non-blocking IO), and why?
- What would need to be done if the Mule runtime could not self-tune these component execution profiles?

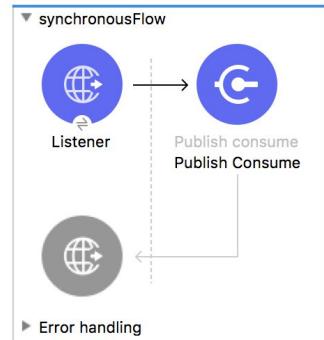
Describing synchronous Mule event processing



Features of synchronous single-threaded processing of Mule flows



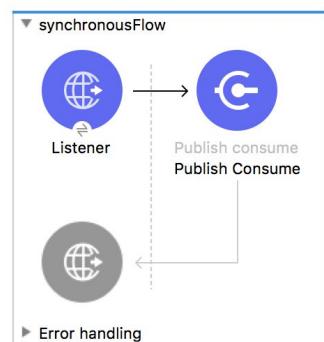
- All processing is performed in a **single thread**
- The main thread **blocks waiting** for the called worker thread to complete execution
- The entire request and response is handled synchronously in **one thread**



How transactions are processed



- **Transactions** are **processed** in **synchronous** mode
 - A transaction scope can be for an entire flow, a Try scope, or an individual connector
 - If a Connector listener operation starts a transaction, the transactional scope is for the entire flow



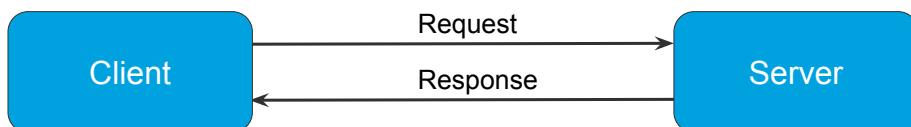
Describing synchronous Mule event processing of web services



Understanding the Hypertext Transfer Protocol (HTTP)



- A **synchronous** and **stateless** communication protocol
- Can transfer **hypermedia** type data
 - Hypermedia is structured text with links to other documents, details, or media
- Built on top of the TCP **client/server** communication protocol
- An HTTP request **message** is submitted from a **client** to a **server**, then the server returns a response back to the client

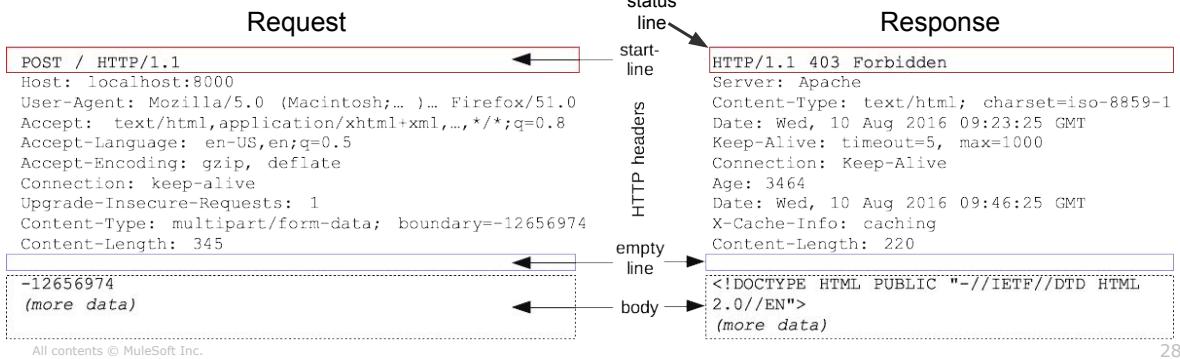


The structure of an HTTP request and response



- The response **status line** with the protocol version, status code, and status message
- **HTTP response headers**
- A **response body** (optional)

- The response **status line** with the protocol version, status code, and status message
- **HTTP response headers**
- A **response body** (optional)



Understanding the Representational State Transfer (REST) protocol



- An **architectural style**
 - To provide **interoperability** between computer systems over the internet
 - consists of a coordinated set of components, connectors, and data elements within a **distributed hypermedia system**
 - Does not specify any underlying communication protocol to use
 - Although in practice RESTful systems **always communicate over HTTP**

Architectural constraints that make a system RESTful



- Addressable resource
 - A resource is any information or concept that can be **named**
 - A resource identifier should **represent only one resource**
- Stateless
 - The server does **not store application state**
 - Application state is to be managed by the client
- Connectedness
 - The server guides the client to change state
 - This is done through **Hypermedia**, and also through headers in the case of HTTP
- Uniform interface
 - **Resources share interface characteristics**
- Idempotency
 - The same **HTTP request** should always create the same server state

All contents © MuleSoft Inc.

30

In practice, RESTful services always use HTTP as a uniform interface



- Resources are operated on using standard HTTP methods
 - Unlike SOAP, this standardizes **C.R.U.D** operation names in every API
- HTTP methods can support RESTful operations
 - In practice, many REST API implementations violate some RESTful principles
 - For example, not using Hypermedia (links) to guide clients across requests

HTTP method	Intended CRUD operation	Comment	Typical response status codes
POST	Create		201 Created / 405 Method Not Allowed
GET	Read	Idempotent	200 OK / 404 Not Found
PUT	Update/Replace	Idempotent	200 OK / 204 No Content
DELETE	Delete	Idempotent	200 OK / 204 No Content
PATCH	Partial update/modify		200 OK / 204 No Content

Reference: <https://www.w3.org/Protocols/rfc2616/rfc2616.txt>

All contents © MuleSoft Inc.

31

Understanding how the SOAP messaging protocol compares with REST



- A specification for **exchanging structured information (messages) via web services** in computer networks
 - Focuses on exposing pieces of application logic (not data) as services
 - Typically used for communication between applications
- SOAP is based on the XML and W3C standard
- Both SOAP and REST support SSL

How REST methods compare with SOAP operations



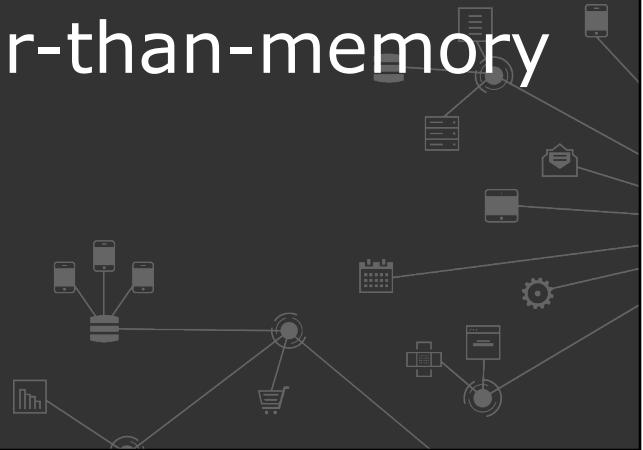
- SOAP focuses on exposing and accessing **named operations**, while REST focuses on accessing resources via standard HTTP methods
 - Unlike REST, operations are not tied to the HTTP method, and can have any arbitrary name
 - All SOAP operations are usually performed via HTTP POST methods
 - Unlike REST, operation names must be invented and typically differ for every SOAP API
 - Each operation usually describes the implementation of some business logic, so the context changes between different SOAP APIs

- Many SOAP extensions begin with the prefix WS-, and are collectively referred to as WS-* standards
- **WS-Security**
 - Adds integrity and confidentiality to SOAP messages using, respectively, **XML Signature** and **XML Encryption**
- **WS-AtomicTransaction**
 - Supports **ACID** Transactions over a service
- **WS-ReliableMessaging**
 - Provides **asynchronous processing** and a **guaranteed level of reliability** by retrying in case of communication failures
 - Provides various levels of **qualities of service (QoS)** for the delivery assurance of messages

Reflection questions

- Why would you develop a new SOAP service instead of a REST service?
- What are the tradeoffs of using SOAP vs. REST?
- How are data schemas validated and enforced in SOAP vs. REST?
- Does REST support XML payloads?
- What types of data formats are supported by SOAP vs. REST?
- What are some other common synchronous data processing models, and what are the tradeoffs compared with REST or SOAP?
- Compare how errors are communicated in REST vs. SOAP vs. other event processing models?
- How are error status messages returned to clients in SOAP vs. REST, and what are the tradeoffs between these two styles?

Streaming larger-than-memory Mule events



Introducing streaming processing models



- **Streams** are data structures that provide efficient processing of large data objects such as files, documents, and records, by processing data **continuously as it arrives**
 - This is **different** behavior compared with other in memory data structures that **hold all values in memory** before computing
 - Allows large datasets to be processed without **running out of memory**
- The Mule 4 runtime **automatically** streams large data payloads without any special configuration
- The default streaming option varies for the Mule EE and Community Edition

- Consumers receive individual cursors
 - Provides repeatable and concurrent random access to repeatable streams
 - Only for Mule 4
- Every component in Mule 4.0 that returns an InputStream or a Streamable collection supports repeatable streams
- Some connectors that support repeatable streams include
 - File
 - FTP
 - DB
 - HTTP
 - Sockets
 - Salesforce

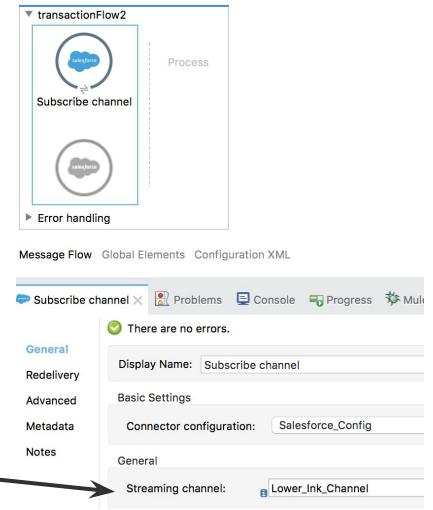
Identifying streaming options for supported Mule 4 connectors

- **File stored repeatable** streams
 - **Default for Mule EE**, and only available in Mule EE
 - By default stores 500 objects in its in-memory buffer
 - This number can be configured
 - Excess objects are serialized using a Kyro serializer that writes to disk
- **In-memory repeatable** streams
 - **Default for Mule Community Edition**
 - Uses a default max buffer size of **500 objects**
 - The buffer is expanded from an initial buffer size (default 100), in a default increment size of 100 objects, **until the max buffer size** is reached
 - If streams exceed the max buffer size, then the application fails
- **Non Repeatable** streams
 - The input stream is only **read once**
 - No extra memory or performance overhead compared with repeatable streams

Describing the Salesforce streaming processing model



- Salesforce provides a **Streaming API**
 - To receive notifications from Salesforce on a PushChannel about modified Salesforce data
 - A PushChannel is defined and customized with a valid Salesforce Object Query Language (SOQL) query
 - Reduces the number of requests that return no data from the Salesforce server
 - By pushing rather than polling for changes
- Mule applications can listen to Salesforce notifications by
 - Subscribing to a Salesforce streaming channel
 - The subscription is handled through a Salesforce connector's **streaming channel** operation configuration

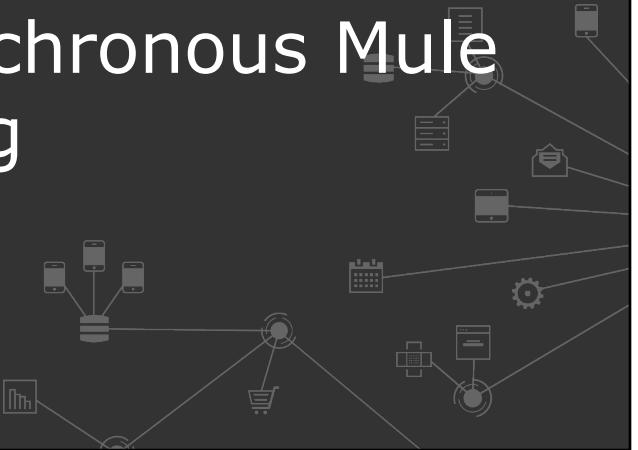


Reflection questions



- What are the tradeoffs between using a file stored repeatable stream (that uses Kryo) vs. an in-memory repeatable stream?
- What type of use cases would benefit from repeatable streams?
- When would a non-repeatable stream be helpful or necessary?

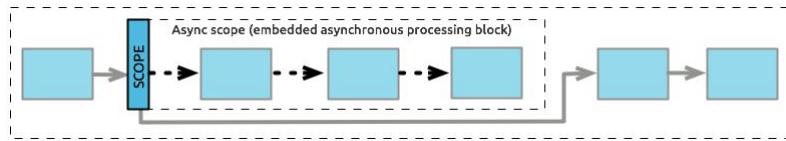
Describing asynchronous Mule event processing



Asynchronous processing models



- The main thread **does not wait** for its worker threads to complete execution
- Uses branch processing that executes separate worker threads simultaneously with the main flow's thread(s)
- **Failure** in one of the branches does **not** impact the main flow
- **Responses** from branch processing are **not** available to the main flow



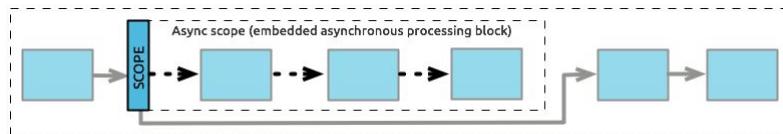
Distinguishing between consumable and non-consumable event payloads



- Mule events can contain consumable or non-consumable payloads
- A consumable payload **cannot be re-read**, so there is never contention between asynchronous processing of the same Mule event
- Non-consumable payloads **can be re-read** between threads, so there could be a race condition between threads or flow routes

All contents © MuleSoft Inc.

44



Mule 4 simplifies side effects from non-consumable payload

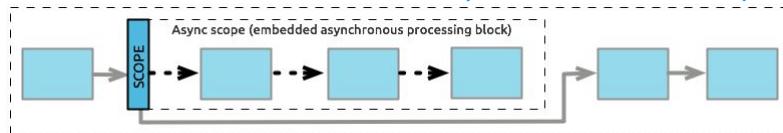


- In Mule 4, The Mule runtime's repeatable streaming hides all the complexity of **sharing Mule events between threads and routes**
- Unlike Mule 3, you don't have to worry about
 - Which components are streaming and which are not?
 - Is the stream consumed at a particular point?
 - At which position is the stream at anyway?
 - What does streaming even mean?
- For rare corner cases, repeatable streaming can be disabled
 - In which case you will need to again worry about these issues

<https://blogs.mulesoft.com/dev/mule-dev/10-ways-mule-4-will-make-your-life-easier/>

All contents © MuleSoft Inc.

45



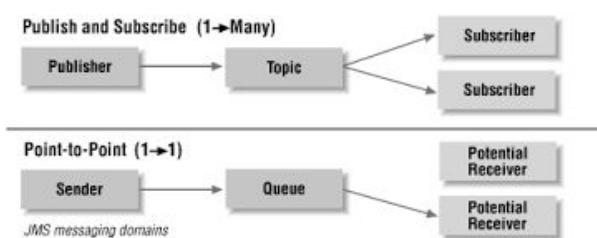
Describing asynchronous Mule event processing using JMS



Understanding the **Java Message Service** (JMS) message processing model



- JMS is a Java standard
 - Specifies how to exchange messages between **producers** and **consumers** through an intermediary **message broker** (also called the **JMS provider**)
- Supports **one-to-one and one-to-many exchanges** of messages
- Messages can be stored and forwarded in the message **broker**
 - To support **asynchronous** message exchange patterns

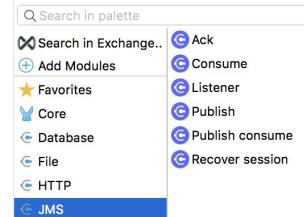


Using a JMS connector in a Mule application



- Connects to an external JMS provider (also called a message broker)
 - The JMS connector relies on a JMS client library for a particular JMS provider
 - This library is added to the Mule application as a Maven dependency
 - Simplifies Mule application development by providing a standard interface for creating **destinations** and exchanging messages through destinations

JMS_1_0_2b
✓ JMS_1_1 (Default)
JMS_2_0

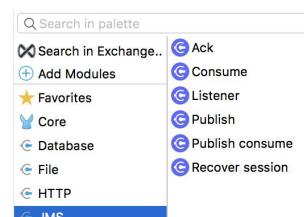


Types of message exchanges supported by an Anypoint JMS connectors



- Supports message exchanges with both types of JMS destinations
 - **Queues** (Point to point messaging model)
 - **Topics** (one to many publish/subscribe messaging model)

JMS_1_0_2b
✓ JMS_1_1 (Default)
JMS_2_0



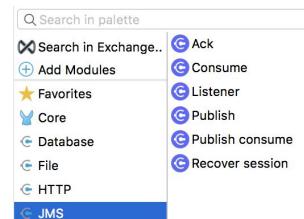
Types of message exchanges supported by an Anypoint JMS connectors



- Supports both **synchronous** and **asynchronous** communication

- Publish sends a message for asynchronous delivery
 - Immediately moves forward to the next event processor and never gets a response back from the JMS provider
- Consume operation performs a blocking receipt of a message, anywhere in a flow
- Listen operation is an Event source that triggers the flow upon receipt of a message
- Publish consume operation is synchronous
 - Blocks the flow until a response is returned from the JMS provider or a timeout expires
 - Combines publish and immediate (blocking) consume operations

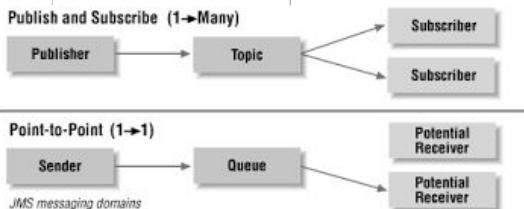
JMS_1_0_2b
✓ JMS_1_1 (Default)
JMS_2_0



Distinguishing between how JMS queues and topics handle messages in asynchronous communication



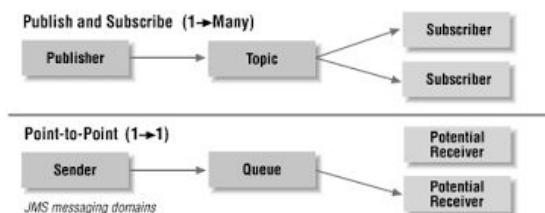
	Queue	Topic
Typical use cases	Work distribution	One-to-many broadcasts
Number of connected consumers allowed	Zero or more	Zero or more
Who receives message	One receiver	All active subscribers
If no active subscribers	Message persists	Message dropped if subscriber is not durable



The order that messages are delivered to consumers from a JMS queue



- The JMS provider queues messages **in the order** they are received
- For a **single consumer**, the JMS provider dispatches messages to the connected consumer, **in order**
- If there are multiple message consumer instances consuming from the same queue (whether in the same JVM or not) then
 - Messages are no longer guaranteed to be processed in order
 - This is because messages will be processed concurrently in different threads or different processes



Describing asynchronous Mule event processing using VM queues

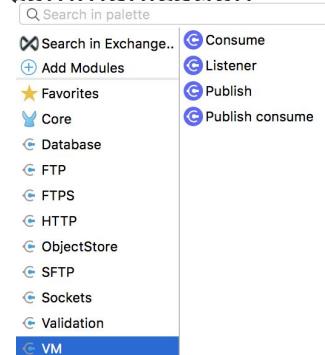


How VM connectors work



- Unlike JMS, VM queues do not use any intermediate message broker
- Creates and communicates with virtual machine (VM) queues using a publish/consume messaging model
- Supports intra-app and inter-app (in Mule domain) communication through queues that can be transient or persistent
- Supports sync or async communication

All contents © MuleSoft Inc.

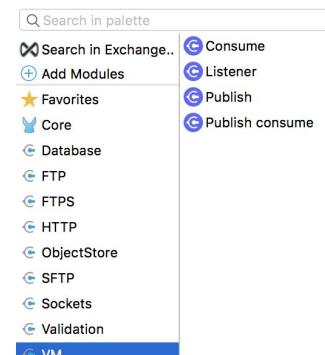


How persistent VM queues are implemented



- On a single customer-hosted Mule runtime instance, persistent VM queues work by serializing and storing the contents on the disk
- In a cluster of customer-hosted Mule runtimes, persistent queues are backed by the Hazelcast distributed data grid
 - Can be configured to be only in-memory or also persisted to disk
- In CloudHub, VM queues are stored in a CloudHub service
 - Persistent queues retain data after the Mule application restarts

All contents © MuleSoft Inc.

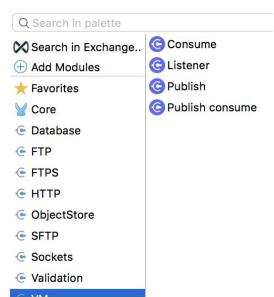


- Mule 4 EE supports a Kryo based alternative to default Java serialization to serialize objects to improve performance in particular use case
 - Read/write from a persistent ObjectStore
 - Read/write from a persistent VM or JMS queue
 - Replicating an object across a Mule cluster
 - Read/write an object from a file
- Note: Kryo is only used by some components (Batch, repeatable streaming) by default, but can be set as the default Serializer throughout the Mule runtime"

<https://blogs.mulesoft.com/dev/tech-ramblings/need-77-performance-boost-no-problem-with-mule-3-7-using-kryo/>

Comparing VM queues with other messaging solutions

- Many Mule applications use an external messaging system like Anypoint MQ or a JMS provider
- With VM queues, reliability and quality of service is limited
- VM queues can be used for specific use cases
 - To distribute messages (work) across a cluster of Mule runtimes
 - For high-performance async communication within a Mule application
 - In CloudHub, to distribute work within the same Mule application across multiple CloudHub workers, perhaps with lower latency compared to other messaging solutions
 - When investment in a JMS broker or other messaging system is not justified or supported

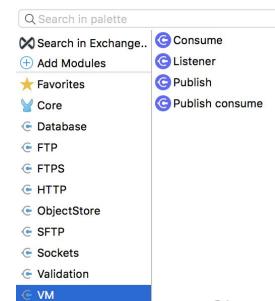


Comparing VM queue load balancing compared to other messaging solutions



- When a Mule application is deployed to a customer-hosted cluster or multiple CloudHub workers, messages are consumed in a fast, non-deterministic way
 - Not strictly round-robin
 - One node (worker) might get a sequence of messages
 - The load balancing algorithm cannot be changed or tuned
- Full messaging solutions are often more flexible
 - May allow the load balancing algorithm to be changed or tuned
 - Often allow messages (work) to be distributed to different applications, even non-mule applications

All contents © MuleSoft Inc.



61

Mule 4 event processing models for asynchronous messaging



- Implemented using specific messaging connectors
 - Such as the connectors for **Java Message Service (JMS)**, **Anypoint MQ**, or **VM**
 - Each of these **external messaging systems** has their own concept of a message and message structure
 - In Mule flows, the messaging structure is transformed to and from a **Mule event**
- Depending on the messaging connector type, messages might be exchanged
 - Through an **intermediate message broker**
 - Or directly in Mule application memory (such as with VM queues)
- Consumers don't need** to be connected at the time the message is produced

All contents © MuleSoft Inc.

62

- Runtime behaviour varies by JMS provider, but is transparent to the Mule JMS Connector
 - Some JMS providers allow a queue to be configured as an **exclusive queue**
 - An exclusive queue guarantees each message is delivered at least ONCE to one of the cluster's consumers
 - Then message processing is automatically bound to one connected consumer, called the **exclusive consumer**
 - This one consumer processes all the queue messages, in order
 - In case of failure of the primary node, another node will be elected as the exclusive consumer and will take over processing the rest of the messages in the exclusive queue

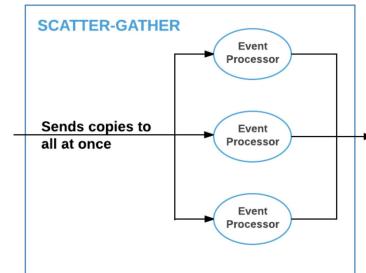
Describing parallel Mule event processing



Describing the Scatter-Gather processing model



- The Scatter-Gather component is a routing event processor
 - Processes the Mule event **in parallel in multiple routes**
 - Parallel execution of routes may **improve performance**
 - Mule 4 repeatable streaming eliminates any possible contention of the Mule event as it is processed through each route
 - Just like in any Mule flow, if an event processor in any route consumes any part of the Mule event, a new Mule event is created for that route
- The Scatter-Gather completes after every route completes, or after a configured timeout expires



65

Iteratively processing collections of records



Processing collections all at once using a connector's batch operation

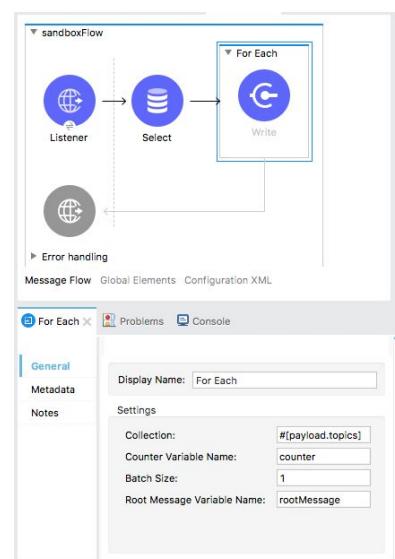


- Some connectors support batch operations
 - Collections of items can be batched together and sent as a single event to the connector

Using For Each scopes to process collections of items



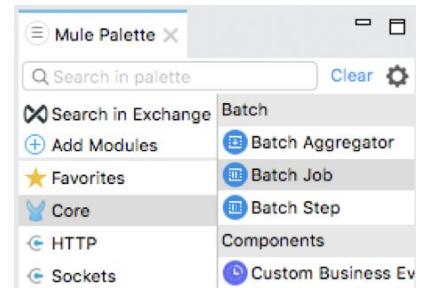
- The For Each scope is configured with a Collection object
 - Synchronously processes one item at a time from the Collection
 - Cannot stream the payload
- The payload after the For Each scope returns to the unprocessed payload before the For Each scope



Process batch jobs asynchronously using a Batch Job scope



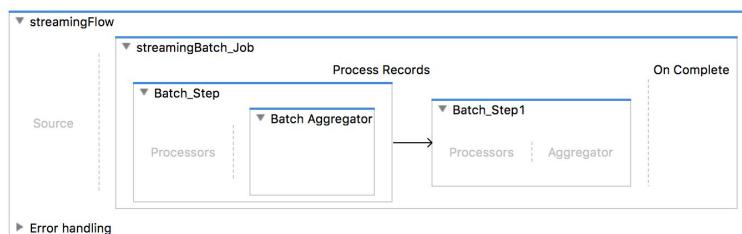
- Provides ability to split large messages into records that are processed **asynchronously** in a batch job
- Created especially for processing data sets
 - Splits large or streamed messages into individual records
 - Performs actions upon each record
 - Handles record level failures that occur so batch job is not aborted
 - Reports on the results
 - Potentially pushes the processed output to other systems or queues
- Enterprise edition only**



Describing the Batch Job processing model



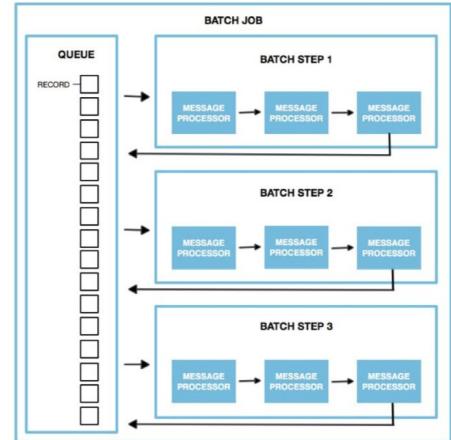
- Only available with Mule Enterprise Edition (EE) runtimes
- Processes individual records from a collection across one or more batch steps, one step at a time
- Provides the ability to split large messages in a "splittable" format such as a collection or an array, streaming or not, into records that are processed asynchronously in a batch job
 - A certain number of records can be aggregated together and sent all together to other systems or queues



How Batch Jobs process records



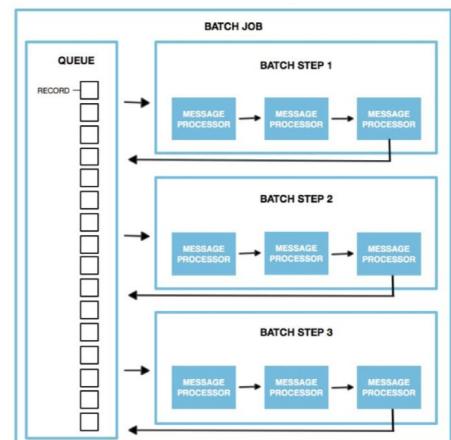
- From the input payload, a **fixed block of records** (a batch) is sent through the batch job for processing by all the batch steps
- All the batch records are first queued
- Records are taken from the top of the queue, one at a time, and sent to the first batch step
- Several threads may process multiple records in parallel



How records can jump ahead to later batch steps



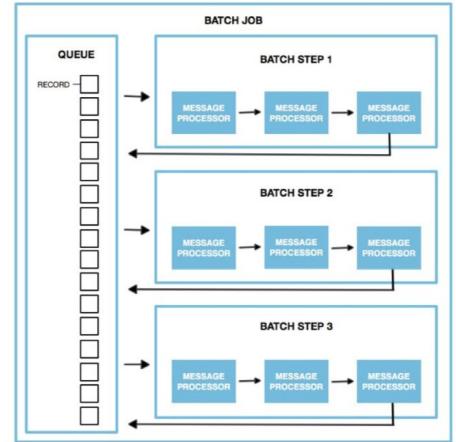
- After completing the batch step, the record is put on the **bottom** of the queue
- Records are always processed, in order, by one particular batch step
- But later records can **jump ahead** to the next batch step while earlier records are still being processed by another thread in the previous batch step
- In this way, some later records in the original record queue may complete traversing through the entire batch job before earlier slower records finish



Filtering which batch steps a record uses



- Batch step filters with **accept Expressions** are used to process records that didn't fail during the previous batch step
- Other filters can configure a batch step to handle previously failed records



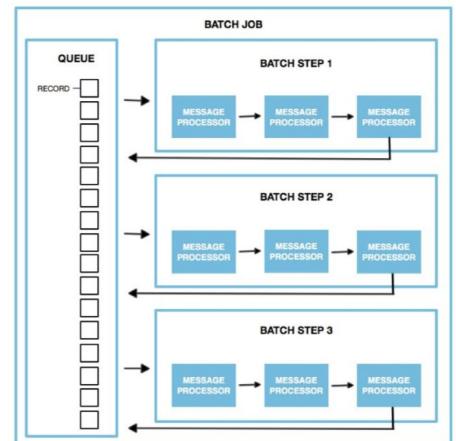
All contents © MuleSoft Inc.

73

How a batch job completes



- Successful execution of a batch job creates a **BatchJobResult** object
 - Contains a summary report about the records processed for the particular batch job instance
 - Does not include any of the actual processed records or data
 - The payload after a Batch Job is the original payload before entering theBatch Job



All contents © MuleSoft Inc.

74

Reflection questions



- What is the difference between using a Batch Job vs. calling a For Each scope that contains an Async scope?
 - Can a record in a Batch Job complete a later Batch Step before finishing a previous Batch Step?
 - Can a later record in a Batch Job complete before an earlier record completes?

All contents © MuleSoft Inc.

76

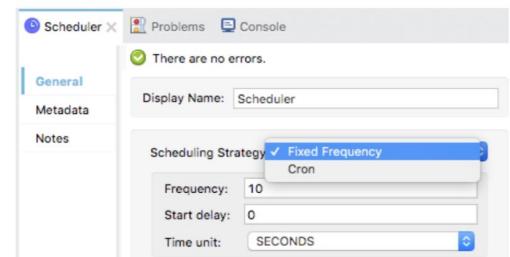
Synchronizing data with real-time and scheduled (batch) Mule event processing



How Scheduler components processing Mule events



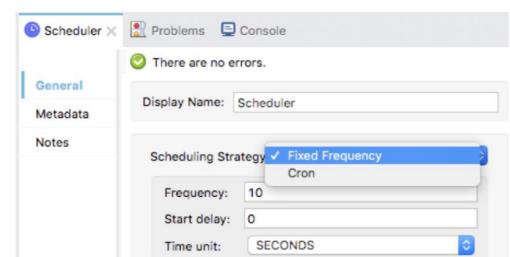
- To trigger any flow **at any time**, use the **Scheduler** component
- In a Mule runtime cluster or multi-worker CloudHub deployment, the Scheduler only runs (or triggers) on one **single** Mule runtime and it is **not guaranteed to always run on the same instance** of that Mule runtime
- Max Concurrency of 1 set on the scheduler flow does not ensure only one instance of the Scheduler runs
 - If the scheduler interval is shorter than the flow execution time



Selecting and configuring a scheduling strategy



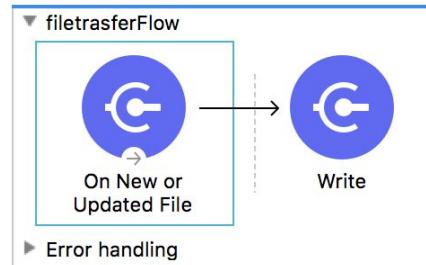
- Fixed frequency
 - Set the time unit and frequency
- Cron
 - A standard for describing time and date information
 - Can specify either
 - An event to occur just once at a certain time
 - A recurring event on some frequency
- A time zone used in a Scheduler corresponds to the region of the one CloudHub worker running the scheduler



Describing the File Transfer Protocol (FTP) processing model



- The FTP connector provides access to files and folders on an FTP/SFTP server
- Supports common FTP operations
 - Listen for new or modified files
 - Read files
 - List directory contents
 - Create directory
 - Copy, move, rename and delete files
 - Write to file
 - Lock files
 - File matching
 - Watermark Enabled



All contents © MuleSoft Inc.

81

Describing the Salesforce connector event processing model



- Salesforce applications send events (or notifications) that Mule applications consume to take further action
- Publishers and subscribers communicate with each other through events
- One or more subscribers can listen to the same event and carry out actions
- Mule applications can listen to event messages by subscribing to a topic
- The Salesforce connector supports event processing by subscribing to a topic

transactionFlow3

Subscribe topic

Process

Message Flow Global Elements Configuration XML

Subscribe topic X Problems Console Progress Mule

General There are no errors.

Display Name: Subscribe topic

Redelivery

Advanced Basic Settings

Metadata Connector configuration: Salesforce_Config

Notes General

Topic: Low_Ink_e

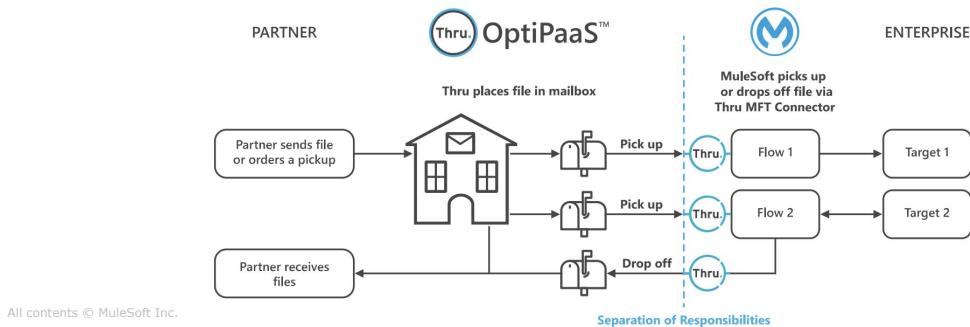
All contents © MuleSoft Inc.

82

Describing the **Thru Managed File Transfer** processing model



- **Thru MFT** is a MuleSoft certified third party connector
 - Free with a Mule EE subscription
 - Download from Anypoint Exchange
 - Provides file management via an Thru's OptiPaaS file management server
 - Provides audits, alerts, and replay for file transfers
 - Include processing dashboards to monitor and control all file transfers



How Thru MFT works



- Thru **OptiPaaS™ (MFTaaS)** is the **post office** where partners and the enterprise set up their endpoints to send and receive files for each process
- The same endpoints can be used in multiple processes and hence referred to as **reusable** endpoints
- The process uses a connector (Thru MFT Connector for MuleSoft, in this example) **to pick up files and possibly drop off files** with delivery instructions
- The post office places files of a specific process in a single **Pick Up Box**
- This means the consuming process (the flow) is decoupled from file sources and does not change when file sources change
- The platform takes the files from the "Drop Off Box" and delivers them to the targets via the organization endpoints

Deciding between Mule event processing options



Abstracting event processing options into processing models



- **Processing models** collect together options and behaviors related to Mule event processing by Mule application components
- Event processing by **Mule scopes** can be modelled to describe
 - How Mule components can process a copy of the same Mule event in parallel, then combine the results
 - How Mule components process a Mule event that contains a collection of records, sequentially or in parallel
- Event processing by **Mule connectors** can be modelled to describe
 - The behavior of inbound Mule event sources at the beginning of a flow
 - How message queuing connectors process Mule events

Deciding between processing models for the use case



- Selecting the best processing model for a use case involves various factors
 - Throughput of data (dataset)
 - Memory or CPU limitation
 - Latency/Response Time
 - Message size
 - Performance requirement
 - Cluster or load balanced deployment
 - Request, response exchange pattern
 - Processing - parallel, sequential
 - Error handling and error response
 - Concurrency requirement
 - Advance capability of connectors or supported connectors
 - Real time, near real time, schedule, periodic processing of data
 - Synchronization of resource
 - Atomicity
 - Idempotency of system

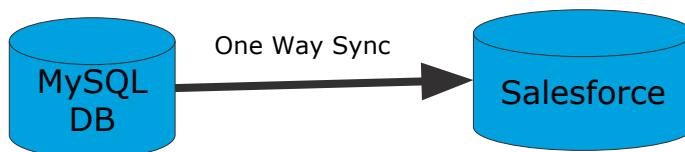
All contents © MuleSoft Inc.

87

Exercise 4-1: Appropriately model Mule event processing for a data synchronization use case



- Choose the most appropriate integration style to design integration solutions that meets the organization's current requirement
- Design an integration solution to synchronize data between two databases or other systems of record
- Modify the integration style (Mule event processing models) based on data throughput vs. data processing latency requirements
- Identify the impact of competing requirements on the selected integration style



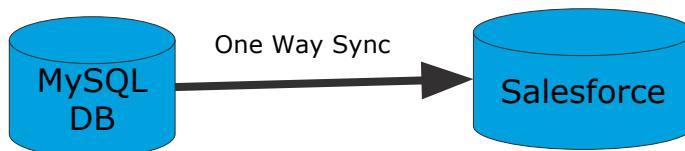
All contents © MuleSoft Inc.

88

Exercise context



- Use case: One way syncing of data between a **source** MySQL database and a **target** Salesforce CRM system or other target database
- SLAs include
 - Average and maximum throughput rates of 200 records per minute



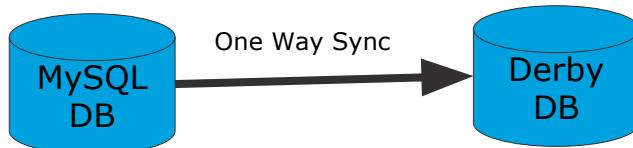
All contents © MuleSoft Inc.

89

Exercise steps



- Decide and document your assumptions of the data processing requirements (SLAs) for the use case
 - Average and maximum throughput rates
 - Processing latency SLAs
 - Must data be processed in real time or near real time?
 - If not, at what rate is data processing scheduled?
 - Will schedule data processing be at a fixed rate, or on specific dates (cron jobs)?
- Document a proposed processing model for the data synchronization use case based on the data processing SLAs

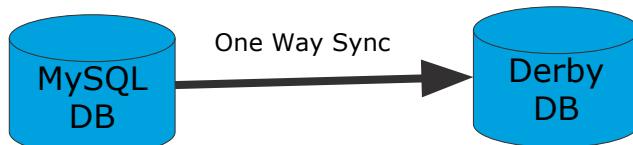


All contents © MuleSoft Inc.

90

Exercise steps

- Analyze data processing requirements and tradeoffs for the use case
 - Decide how the processing model can change based on changes to these competing data processing SLAs
 - Average and maximum throughput rates
 - Processing latency SLAs
- Explain how these factors impact your processing model

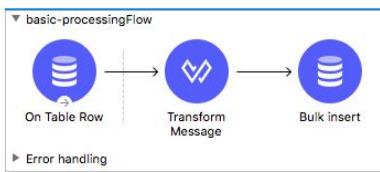


Exercise steps

- Answer these additional questions to decide and verify your proposed event processing model
 - How is data read from the DB?
 - How is flow processing triggered?
 - Must the DB reading option change to support real time or near real time latency SLAs?
 - Does the number or size of records read impact how records are processed?
 - How can the data synchronization between the system be limited to only new or modified records?
- What MuleSoft tools can help you validate your proposed processing model?

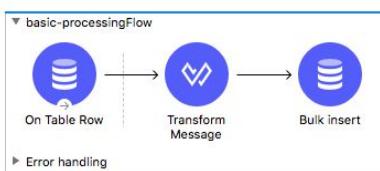
Exercise solution

- One solution is proposed in a sketched Mule flow
 - On Table Row as a message source
 - On Table Row message source in first flow allows to perform **real time** processing from the source MySQL DB
 - Use Bulk insert to improve performance



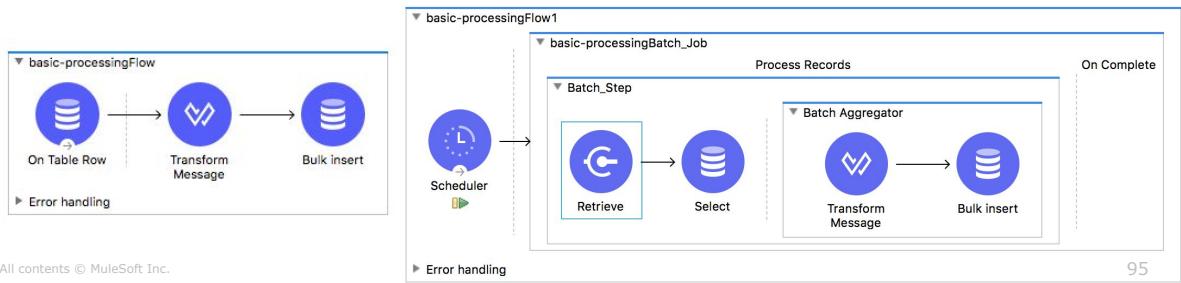
Exercise solution

- What is another way to poll the source database records periodically?
- What are the tradeoffs to using Bulk insert vs. individual inserts?
- What is the tradeoff of using On Table Row vs. a Scheduler and a Select operation?
 - Which one of these triggers can set streaming data?



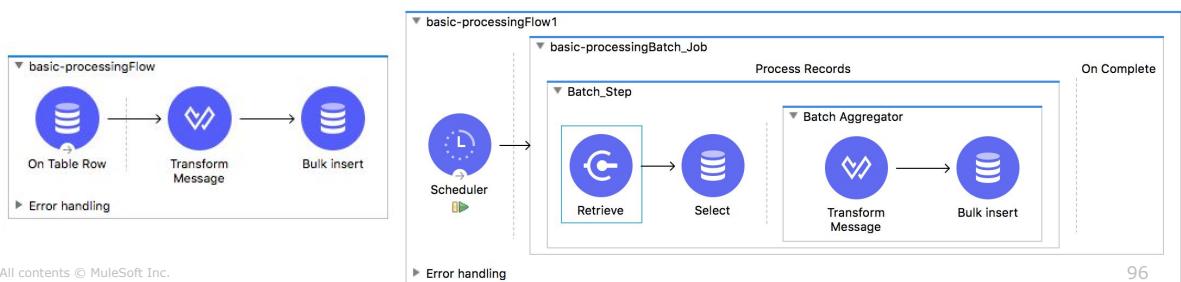
Exercise solution

- A second proposed solution is also sketched in a second Mule flow
 - The second flow uses a Scheduler as the message source
 - The scheduler performs **scheduled** processing from the source MySQL DB
 - The Batch Step includes a Batch Aggregator scope to collect records together that are sent all together to the Bulk insert operation
 - What are the tradeoffs to using bulk operations vs. processing individual records?



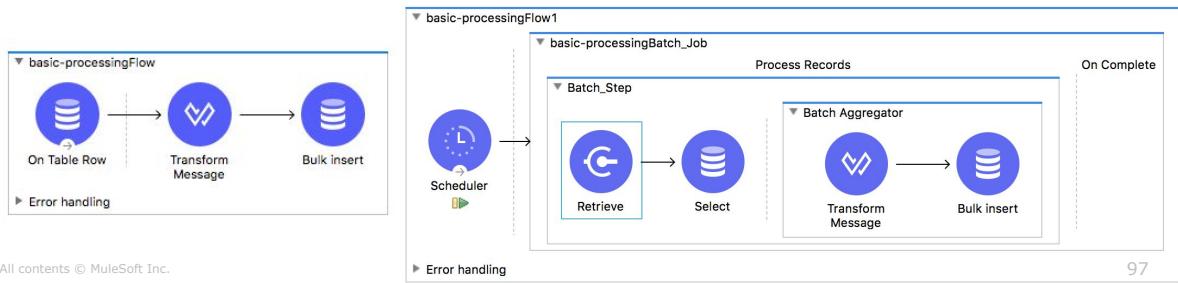
Exercise solution

- Which components should have streaming configured?
- What are the tradeoffs to enabling streaming?
- What are the design and performance tradeoffs between
 - Using streaming (in-memory or in a file) and no Batch Aggregator or Bulk operation
 - Not using streaming, with a Batch Aggregator and a Bulk operation



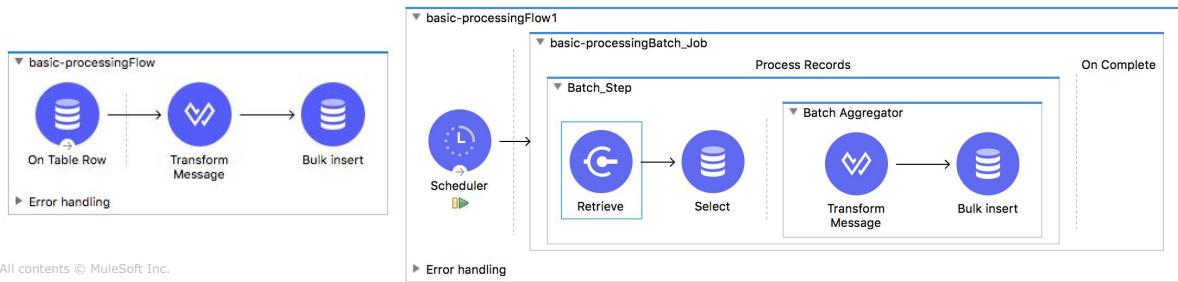
Exercise solution

- Comparing data throughput in both processing models
 - Throughput is comparatively larger in the second flow due to periodic polling without a watermark
 - Throughput is initially the same in both flows the first time data is retrieved from the database, but varies for subsequent selects from the database



Exercise solution

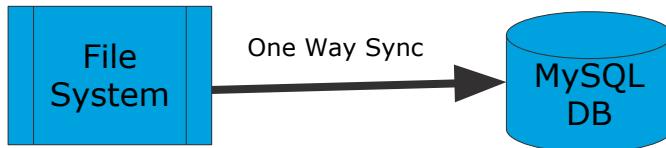
- Comparing configurations of both the flows
 - Watermarking (last record processed) is managed by On Table Row while a Scheduler is used to maintain the watermark
 - Bulk insert is performed in both the flow to commit data in batch



Exercise 4-2: Appropriately design Mule event processing for a file transfer use case



- Identify and recommend the Mule event processing models for a file transfer use case
- Identify and explain every factor that helps evaluate the best processing model
- Justify each Mule event processing model decision based on the identified factors
- Document flows that can implement the selected Mule event processing models



Exercise context: File transfer use case requirements and constraints



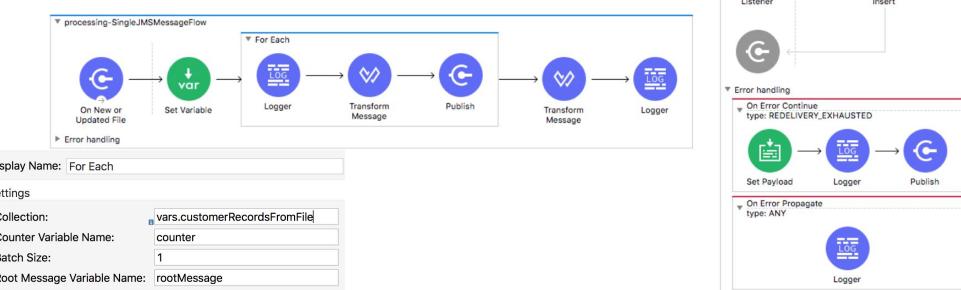
- Randomly, **a few times a day**, a **new source file** containing financial data for customers are **uploaded** to a specific directory on the **file server**
- The Mule application must quickly **process the file** and then send results to a **target MySQL database**
- Each file has about **1 million records** of customer data
- **Auditing** and **traceability** of each record is critical
- **Human intervention** must be allowed to **post process** any **failed records**
- The Mule application has been budgeted to deploy to **one 0.2 vCore CloudHub worker** (with 1 GB of heap memory)

Exercise steps

- Define options to process the file
 - What are the options to read in a file?
 - After reading in a file, how are records in files processed?
 - Should records be processed sequentially, parallelly, or in batch?
 - How can the memory footprint be reduced while processing?
 - How are failed records managed?

Exercise solution

- A proposed processing model using a For Each scope
 - Processing is sequential, even though records are published to a JMS destination
 - For Each does not support buffering so the entire file is loaded into memory
 - Throughput of the process is determined and limited by the For Each scope
 - A separate insert into the DB is performed for each record

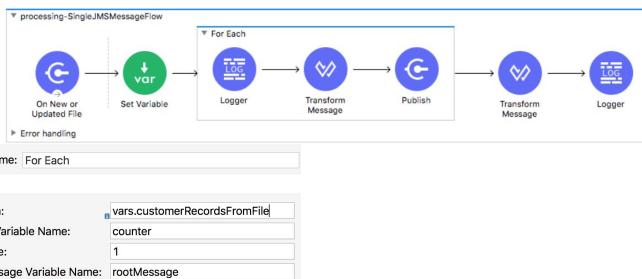


Exercise solution: Processing tradeoffs when using a For Each scope



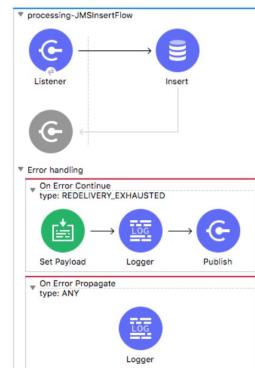
Pros

- Audit and tracing per record
- Auditing and tracing on records is easier and more open with a JMS topic
- Easier and isolated error handling of failed inserts to the target DB



Cons

- Throughput limited by the For Each scope
- Difficult to identify when file processing completes

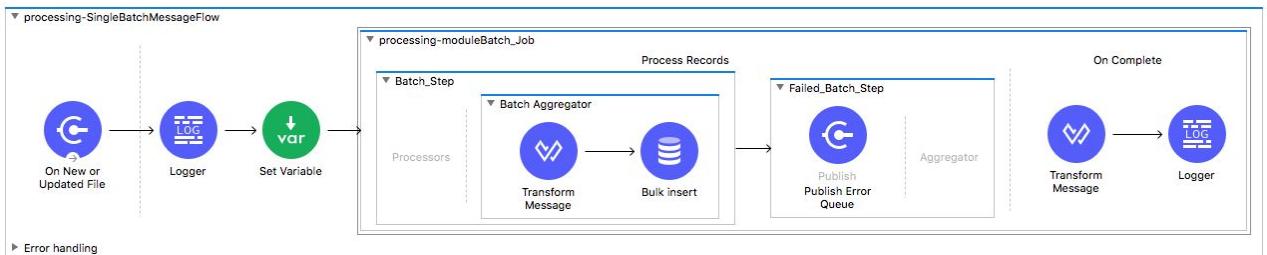


103

Exercise solution



- A different proposed processing model using a Batch Job scope
 - File is read as a stream in a Batch Job scope
 - The stream is transformed in a Batch Aggregator and then multiple records are inserted into the target DB in a single Bulk Insert operation
 - Failed records are sent to a JMS server to a dead letter queue (DLQ)



Exercise solution: Processing tradeoffs when using a Batch Job scope

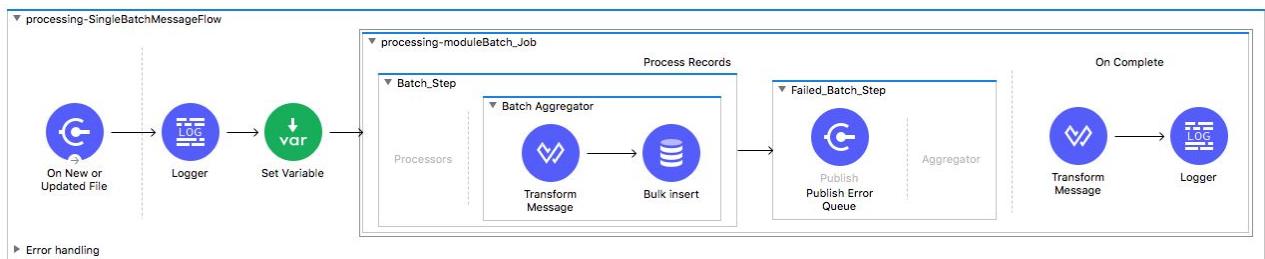


Pros

- Audit and tracing per record
- ~~Auditing and tracing on records is easier and more open with a JMS topic~~
- ~~Easier and isolated error handling of failed inserts to the target DB~~

Cons

- Uses internal queues
 - May cause out of memory errors with large payloads and high throughput



All contents © MuleSoft Inc.

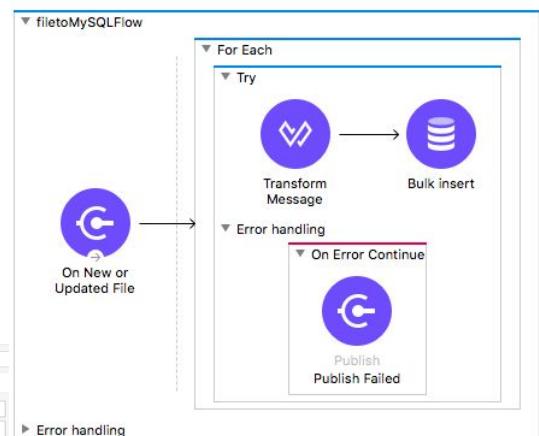
105

Exercise solution



- A proposed processing model using a For Each scope configured to process the source file as streaming data
 - File is read as a stream in the flow
 - The stream is transformed using DataWeave inside a Try Catch block
 - Then records are inserted into the target DB in batch commit using the batch size of For Each
 - The batch size can be tuned by Ops as a property placeholder
 - Failed records are sent to DLQ in the On Error scope

Display Name: For Each	
Settings	
Collection:	vars.customerRecordsFromFile
Counter Variable Name:	counter
Batch Size:	\$db.batchSize
Root Message Variable Name:	rootMessage



All contents © MuleSoft Inc.

106

Exercise solution: Processing tradeoffs when using a For Each scope with streaming data

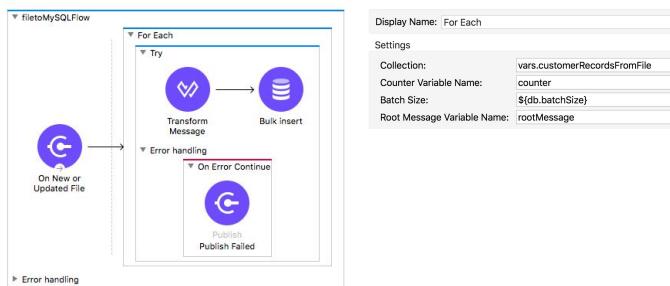


Pros

- Audit and tracing per record
- Audit and tracing on records is easier with a JMS topic
- Easier and isolated error handling of failed inserts to the target DB
- Database impact is tunable

Cons

- ~~May cause out of memory errors with large payloads and high throughput~~
- ~~Throughput limited by the For Each scope~~
- ~~Difficult to identify when file processing completes~~



107

Exercise solution



- Decide the best processing model for the file transfer to database synchronization use case

Factors	Optimum processing model
Large payload	<ul style="list-style-type: none">• Payload has 1 million records• Streaming is the best option for effective utilization of memory
Memory/CPU	<ul style="list-style-type: none">• Limited size of vCore requires the processing model should work with smaller memory and CPU footprints• Streaming is the best option

- Documented flow

<https://docs.google.com/document/d/1C9Xd75I4qO8yGYXIW8sFshSuuiHiZoY19q2iN2Tmync/edit>

Summary



Summary



- Selecting the best processing model for use cases involve various factors
- Mule provides different options to process large vs. small datasets, streaming or not, sequentially or in parallel order
- A Scatter-Gather component provides parallel processing of events
- Batch processing is exclusive to Mule EE runtimes
- Mule 4 runtimes automatically configure event processors to automatically switch to streaming to handle large datasets

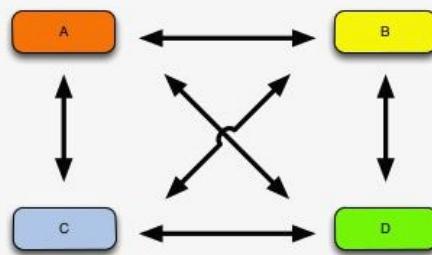
- Salesforce event processing
 - https://developer.salesforce.com/docs/atlas.en-us.platform_events.meta/platform_events/platform_events_intro.htm
- WS-ReliableMessaging
 - <https://en.wikipedia.org/wiki/WS-ReliableMessaging>
- XML Signature
 - https://en.wikipedia.org/wiki/XML_Signature
- XML Encryption
 - https://en.wikipedia.org/wiki/XML_Encryption

Module 5: Choosing Appropriate Message Transformation and Routing Patterns



Goal

$$\frac{n(n-1)}{2}$$



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



- Recognize different integration problems and solve them with common integration patterns that involve Mule applications
- Apply integration patterns to an integration scenario using Mule applications

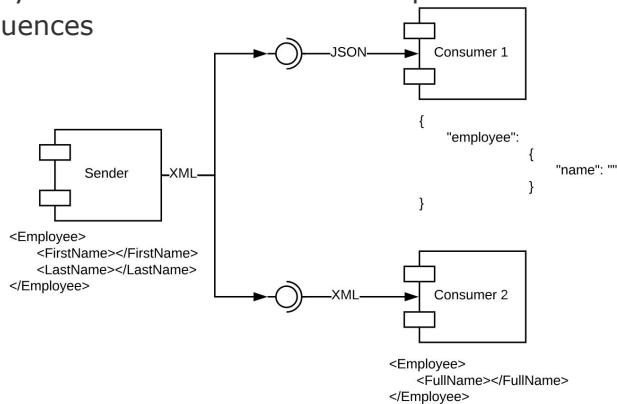
Identifying reusable ways to transform and process events



Why are event transformation and routing patterns needed?



- To identify **reusable ways to transform and process events** between particular enterprise systems
 - Enterprise systems use particular message formats and message schemas
 - Enterprise systems often define business processes that must execute in certain sequences



All contents © MuleSoft Inc.

5

Common message construction patterns implemented in Mule applications



- Common data models
- Message transformation patterns
- Message validation patterns
- Message routing patterns

All contents © MuleSoft Inc.

6

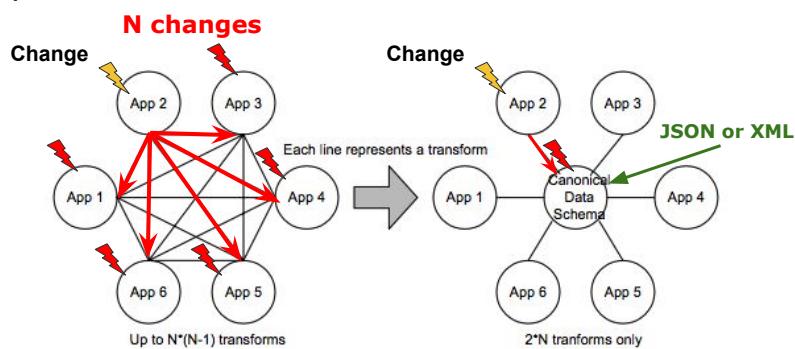
Converting between data models



Transforming between data models



- With or without a common data model, your Mule applications need to **transform** between different data representations
- MuleSoft recommends using a flexible data representation language to develop and maintain application integrations
 - Should be easy to represent and transform between **XML, JSON, Java POJOs, and flat files**



How MuleSoft typically supports modern data types



- Mule application designs and implementations can model data using
 - JSON and XML schema
 - REST API Modeling Language (RAML) data types
 - OpenAPI Specification (OAS) data types
- Both of these schema languages can represent JSON, XML, and other types of objects or flat files
- Anypoint platform supports some interoperability between OAS and RAML
 - Full interoperability is planned in future releases

All contents © MuleSoft Inc.

9

Converting between popular API specification formats



- There are online tools to convert between various API specification formats
 - <https://apimatic.io/>
 - This is helpful to convert other formats to RAML/OAS
 - So they can be imported into API designer

Inputs	Outputs
API Blueprint	API Blueprint
OpenAPI/Swagger 1.0 - 1.2	OpenAPI/Swagger 1.2
OpenAPI/Swagger 2.0 JSON	OpenAPI/Swagger 2.0 JSON
OpenAPI/Swagger 2.0 YAML	OpenAPI/Swagger 2.0 YAML
WADL - W3C 2009	WADL - W3C 2009
WSDL 1.1 - W3C	WSDL 1.1 - W3C
Google Discovery	RAML 0.8 - 1.0
RAML 0.8 - 1.0	APIMATIC Format
I/O Docs - Mashery	Postman Collection 1.0 - 2.0
HAR 1.2	OpenAPI Spec 3.0 (JSON)
Postman Collection 1.0 - 2.0, 2.1	OpenAPI Spec 3.0 (YAML)
APIMATIC Format	
Mashape	
OpenAPI Spec 3.0 (JSON)	
OpenAPI Spec 3.0 (YAML)	

All contents © MuleSoft Inc.

10

Choosing event transformation patterns for Mule applications



Why event transformation patterns are needed



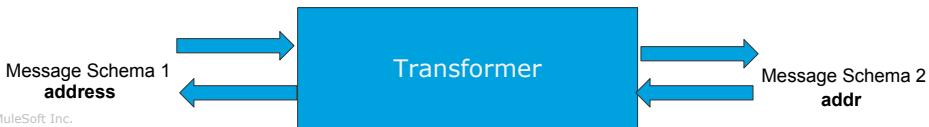
- Different applications, whether homegrown or from different vendors, can use **different** event **payload** schema to represent even the **same** data
- For example, one application can name a customer attribute **address**, whereas the other may use **addr** to represent the same address.



The problem of multiple event transformations



- Schemas used by different systems might contain a **different** number of **fields**
 - And the corresponding fields may differ
 - Such as by having
 - Different field **names** (address vs. addr)
 - Different **types** of value (String vs. Integer)
- To integrate systems, each system's data model must be transformed to the other systems' data model



All contents © MuleSoft Inc.

13

How **DataSense** makes DataWeave mapping easier



- DataWeave supports **DataSense**
 - When connectors supports DataSense, DataSense can **automatically sense and import metadata** from the connectors to help you visually drag and drop DataWeave mappings between input sources and output targets
 - You can also **import schemas** and **sample documents** to more quickly and easily design your transformations, and assist DataSense

American JSON to Flight Java objects

Output Payload

Select metadata type

Choose metadata type from tree and click Select

Add Delete

Type: JSON

Schema: schema/json/americanFlight.json

Variables

Attributes: Unknown

Context: payload

14

Leveraging **DataSense** to implement the transformer pattern



- With DataSense, inbound and outbound schemas can be **auto-populated** in the transformation mapping tools
 - DataSense assists you at **design time** by providing a live stream of content types while you are coding
 - Provides a **scaffolding** to begin writing mappings
 - But not required, the entire DataWeave code can also be typed out manually

```
Output PayLoad - #. #. #. #. 1 issue found
1  " +
2  "    @com.mulesoft.tutorial.Payload<Object>
3  "    destination: payload01.destination,
4  "    flightCode: payload01.code,
5  "    origination: payload01.origin,
6  "    planeType: payload01.plane.type,
7  "    price: payload01.price
8  "
9  ---+
10 @com.mulesoft.tutorial.FlightArray<Object>
11   as Flight
12   flights: [
13     {
14       airlineName: String?
15       availableSeats: Number?
16       departureDate: String?
17       destination: String?
18       flightCode: String?
19       origination: String?
20       planeType: String?
21       price: Number?
22     }
23   ]
24   flights: [
25     payload01.map (payload01, indexOfPayload01) -> {
26       airlineName: airlineName,
27       availableSeats: payload01.emptySeats,
28       departureDate: payload01.departureDate,
29       destination: payload01.destination,
30       flightCode: payload01.code,
31       origination: payload01.origin,
32       planeType: payload01.plane.type,
33       price: payload01.price
34     }
35   ] as Flight
36   flights: [
37     class : "com.mulesoft.training.FlightArray"
38   ] as Object {
39     class : "com.mulesoft.training.Flight"
40   }
41   flights: [
42     class : "com.mulesoft.training.Flight"
43   ]
44   price: 676.0 as Number {class:
45     "double"}
46   flightCode: "euro903" as String
47   {class: "java.lang.String"}, availableSeats: 1 as Number {class:
48     "int"}
49   planeType: "Boeing 737" as String {class: "java.lang.String"}, departureDate:
50   "2015-07-01T12:00:00Z" as String {class: "java.lang.String"}, origination: "MIA" as String {class: "java.lang.String"}, airlineName: "American Airlines" as String {class: "java.lang.String"}, destination: "JFK" as String {class: "java.lang.String"} }
```

15

DataWeave supported formats as input or output

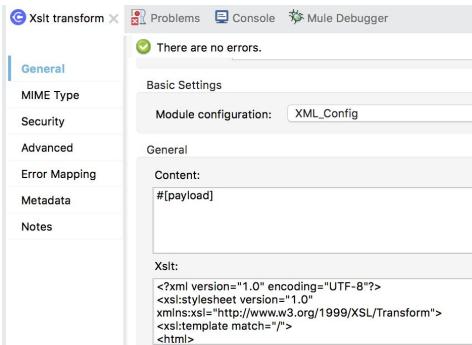


Mime Type	Supported Formats
application/csv	CSV
application/dw	DataWeave (for testing a DataWeave expression)
multipart/*	Multipart (Form-Data)
application/flatfile	Flat File, Cobol Copybook, Fixed Width
application/java	Java, Enum Custom Type (for Java)
application/json	JSON
application/octet-stream	
application/xml	XML, CData Custom Type (for XML)
application/xlsx	Excel
application/x-www-form-urlencoded	URL Encoding
text/plain	For plain text

Using the XML Module's XSLT transformer



- Uses the Extensible Stylesheet Language Transformations (XSLT) language
 - Transforms XML documents to other **XML schemas** or other non-XML formats
 - **XPath** is used to navigate to document structure
- An alternative to using DataWeave transformations



All contents © MuleSoft Inc.

17

Leveraging other flows and services from within a DataWeave expression



- DataWeave can also access other flows, external services, or Java libraries
- This allows some transformation work to be offloaded from DataWeave to other flows or libraries
 - For example, DataWeave can call out to XSLT transformation flows that use the XML Module
 - Or to a different Java library or script that can handle XSLT transformations

18

- Java is generally NOT recommended for data transformation
 - Use DataWeave instead
- However, an organization may want to call out to Java to **reuse a Common object model** that is written in Java and is standard in the organization
 - DataWeave can call out to **static** methods in Java
 - Java classes can be defined in a **Spring context** (in the Mule configuration file or in a separate Spring context file) and invoked from a Java component
 - MuleSoft recommends that you encapsulate custom Java transformation into **classes** that can be **injected** and easily **tested**
 - Mulesoft promotes **separation of concern** hence removed the Expression component and Expression transformer from Mule 4

Transformation using scripting Groovy, JRuby (Ruby), JPython(Python), Nashorn(JavaScript)

- Using supported scripting languages is generally NOT recommended for data transformation
 - Use DataWeave instead

- When is DataWeave a good fit for data transformation?
- Is DataWeave more or less useful when used for CDM mapping?
- What other data transformation options are useful to your Mule applications, and what are the tradeoffs?
- What are the pros and cons of using XSLT vs. DataWeave, and what use cases are especially well suited for XSLT?

Reflection question

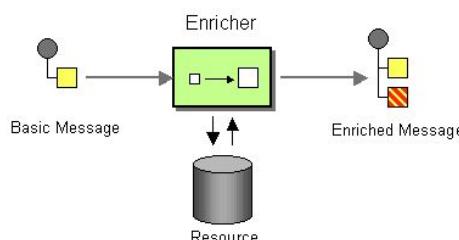
- What are the pros and cons of using custom Java code vs. DataWeave, and what use cases are especially well suited for custom Java code?
- What are the pros and cons of using an external service vs. DataWeave, and what use cases are especially well suited for custom Java code?
- What caching considerations would relate to using external mapping services?
- What evidence can you gather to validate your assumptions about these pros and cons?

Implementing **event enrichment** patterns (also called content enricher patterns)



- Context
 - When an event is sent from a source system to a target system, the target system may require **more information** than the source system can provide

- Problem
 - Communication with the target system **may not work** if the event originator does not have all required content in the event
 - For example, security headers (access tokens) or other required data or attributes



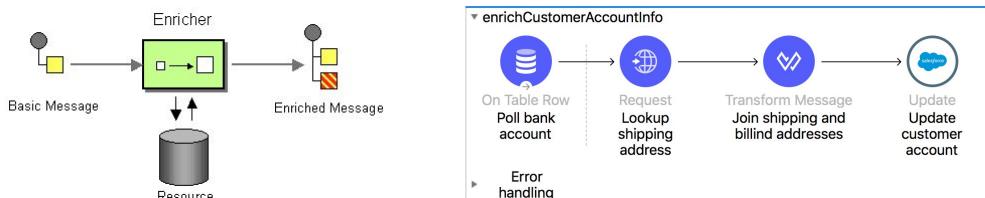
All contents © MuleSoft Inc.

24

Implementing the message enricher pattern with MuleSoft



- Implementation
 - Mule event processors uses information inside the incoming event (e.g. key fields) to retrieve data from an external source
 - After the event processor retrieves the required data from the resource, it appends the data to the payload, or other parts of the message/event
 - Typically the **previous event payload must be stored in a variable, then merged with the new event payload** after the retrieving event processor completes



All content

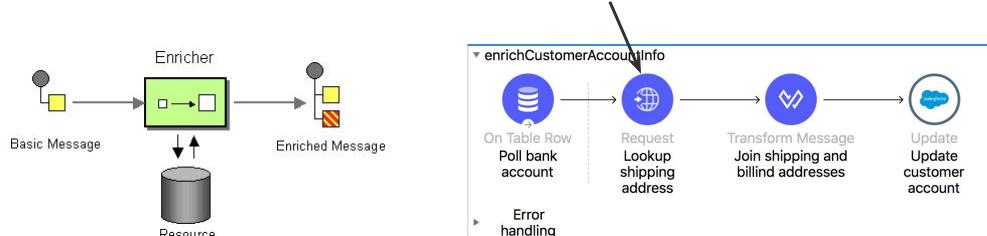
25

Ways to preserve the previous event payload after an event processor executes



- Many event processors have a **target**
 - A target saves you the step of having to store the previous event payload using a Set Variable event processor
 - When a target is set, the event processor result is stored in the variable named by the target
 - The payload from the previous event processor is then left unchanged
 - The target sets its value using a DataWeave expression that can operate on the event processor result (such as the response to a lookup operation)

```
<https:request target="shipAddress" targetValue="#[payload.address]" .../>
```



26

Reflection questions



- How can data be joined between previous event processors, and what are the tradeoffs?
- How does data volume, request volume, or latency concerns affect these tradeoffs and choices?
 - You will see more about these tradeoffs later in this module
- Note: CDMs are discussed in more detail in the Anypoint Platform Architecture: Application Networks course

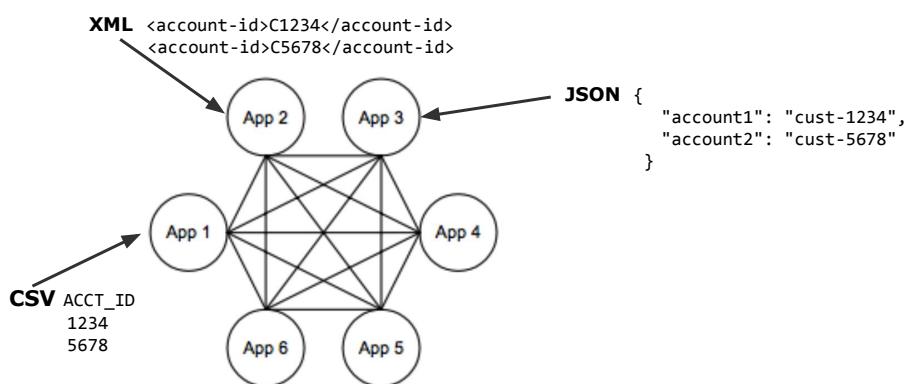
Simplifying and reusing data mappings using common data models



Critical data is often locked up across an enterprise in incompatible silos



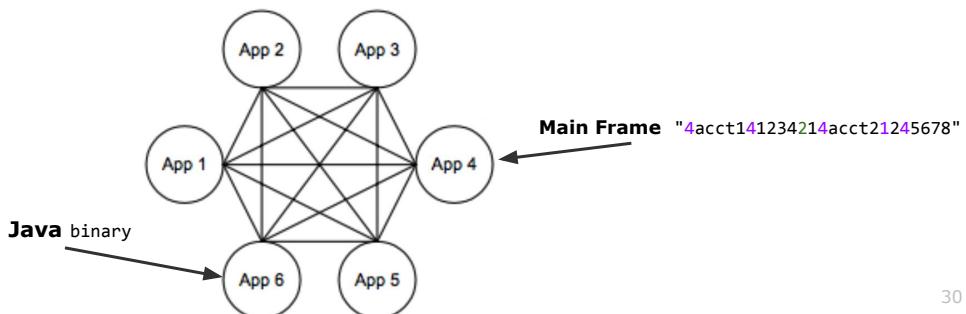
- When an application is integrated with other applications, it needs to know how to
 - Parse data coming in from other applications
 - Format data to be sent out and to be understood by other applications



Why separate point-to-point schema mappings are difficult to build and to maintain



- Ideally, data schema of every app is **known in advance**
- But there can be issues
 - How should data schemas be described and shared?
 - Data represented in one data schema in one system may not be easily transformed into data using another data format and schema type
 - Applications may create **non-standard schema**
 - Data may be a **binary stream**



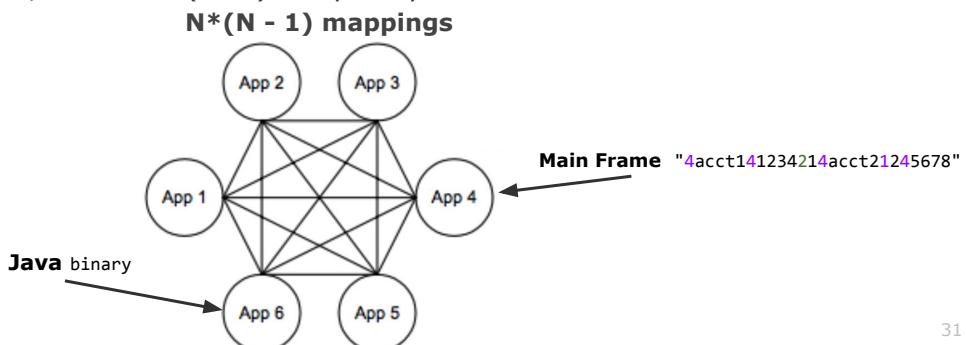
All contents © MuleSoft Inc.

30

The schema mapping explosion problem



- Some mappings may require coupling **two non-standard** and **obsolete** schemas
 - These brittle mappings are difficult and expensive to maintain
- And there are a lot of them
 - In the worst case, $N*(N-1)$ separate and often custom mappings are required, which is $O(N^2)$ complexity



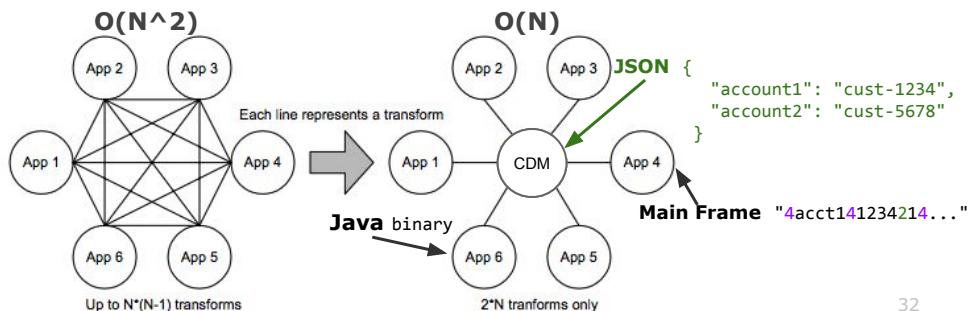
All contents © MuleSoft Inc.

31

The solution provided by a common data model (CDM)



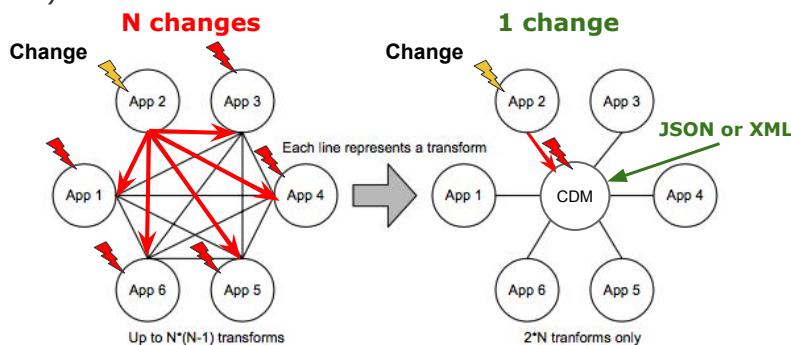
- Build a **common** (also called canonical) data model
 - Specify general rule or acceptable procedure
 - Map from different native application data schema into more common or more neutral data schema
 - In the **worst case**, **$2 \times N$ mappings are required** to support n systems with disparate data schemas, which is $O(N)$ complexity
 - Moreover, each mapping is into an agreed, common, and more open format



A common data model decouples multiple transformation steps



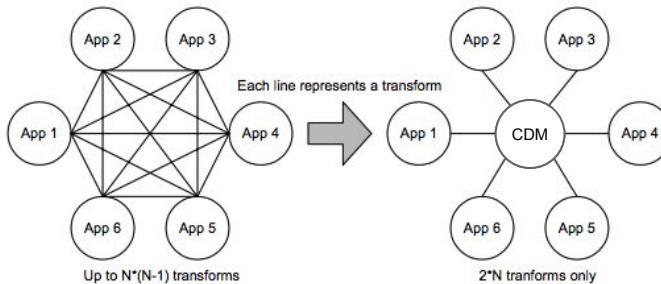
- Isolates backend systems and supporting Mule applications from change
- Encourages innovation and reuses with existing services
 - Encourage evolution of legacy systems and business processes towards more modern, standardized, and cloud-friendly data formats, like JSON (or even XML)



When should a common data model be used?



- To represent data in new applications that need to **communicate with other internal applications** inside the intranet environment
- When disparate organizations or business units can **agree** on a **common** standard or format, or when the scope is narrow enough
- When the process of defining a common data model does **not delay** implementation indefinitely



All contents © MuleSoft Inc.

34

Scoping a common data model



- There are tradeoffs on specifying CDMs depending on the intended scope
- Possible scopes include
 - Just to the **current project**
 - To the **business unit** or other organization context
 - To the **entire enterprise**
- Enterprise-wide CDMs may not be helpful, and may significantly delay implementations
 - It may be difficult or impossible to get agreement across business units or even within one business unit
- Note: CDMs are discussed in more detail in the Anypoint Platform Architecture: Application Networks course

All contents © MuleSoft Inc.

35

- What are the overall pros and cons of creating a CDM?
- What are some scenarios where a common data model (CDM) is an obvious fit?
- What are some scenarios where a CDM is not a good fit?
- At what project or organizational scope(s) are CDMs typically successfully applied?
- At what project or organizational scope(s) are CDMs often a hinderance or liability?
- What are the tradeoffs to building one unified CDM for the entire enterprise (such as a Master Data Management effort)?

Choosing data validation patterns for Mule applications



The problems addressed by message validation patterns

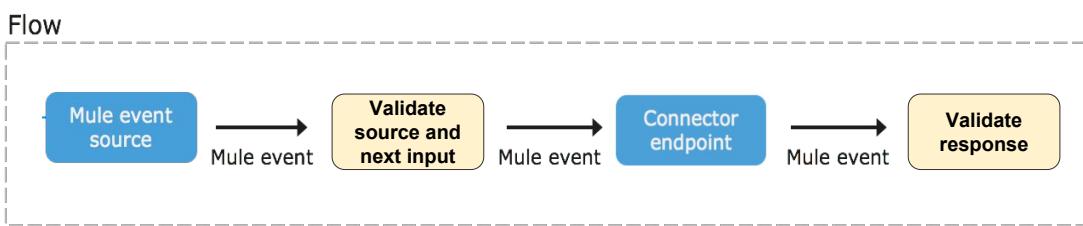


- Context
 - In a flow, the current Mule event may **contain invalid or incorrect data** that was generated by humans, applications, or during transmission
 - Especially after a response from an external system is returned by a connector
 - The current Mule event may also have data that is **incompatible with data** transformation or connector requests later in the flow
- Problem
 - Incorrect data sent to downstream systems may cause data related **errors**
 - At a minimum this could consume **unnecessary system resources**
 - Even worse, unexpected data errors could **crash** downstream applications

How message validation patterns make Mule applications more predictable and reliable



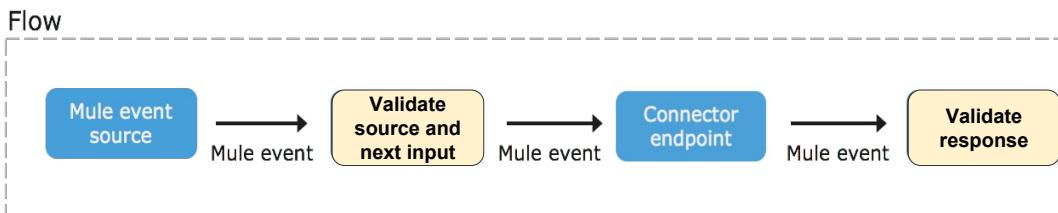
- Solution
 - Mule applications need to **validate** data structures and content as Mule events are transformed through Mule flows
 - Data validation involves some type of **boolean logic test** to decide if particular **expected conditions** are **true or false** at that point of the flow
 - In particular, Mule applications should **identify invalid or incorrect data** and **handle them**, before sending to external downstream systems



Where data should be tested and checked in Mule flows



- In practice, MuleSoft recommends validating events as early as possible
 - At the **beginning of each flow**, if the event source has incoming data
 - Such as for an HTTP listener, database On Table Row, or JMS listener operation
 - Later in a flow, immediately after an **event is created or enriched**
- In other words, your Mule application should **fail fast**



All contents © MuleSoft Inc.

40

Ways to implement data validation patterns in Mule applications



- Common ways to validate events include
 - **Choice routers**
 - Call different flows based on the previous event processors response
 - **DataWeave code**
 - In any Mule component, write boolean logic to test data or conditions
 - In a Transform Message component, set variables based on results of boolean logic tests
 - **Validation modules**
 - Each validation operation throws an error when a data validation condition is false
 - Core Validation module
 - XML and JSON schema validators
 - APIkit validation
 - HTTP Request success and failure status code validators
 - **Catch and handle errors**
 - If possible, recover from validation errors and continue event processing

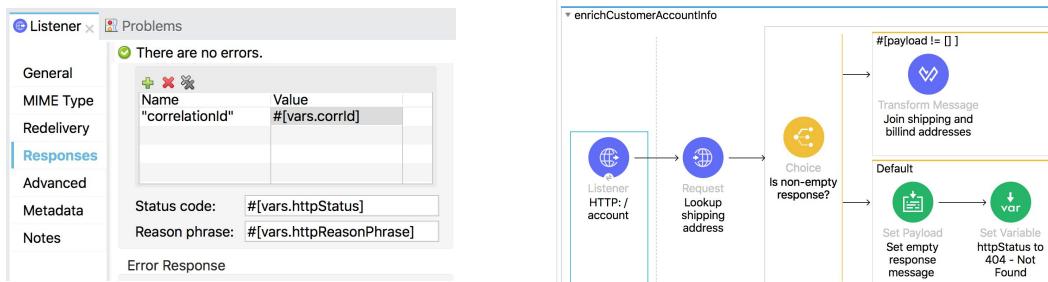
All contents © MuleSoft Inc.

41

Validating and routing response events



- Data validation results can be stored in variables
- HTTP Listener responses can be set from these data validation variables
- A Choice router supports flow control logic to route the Mule event
 - Can call different flows based on the previous event processors response
 - The routing logic can include event validation DataWeave expressions

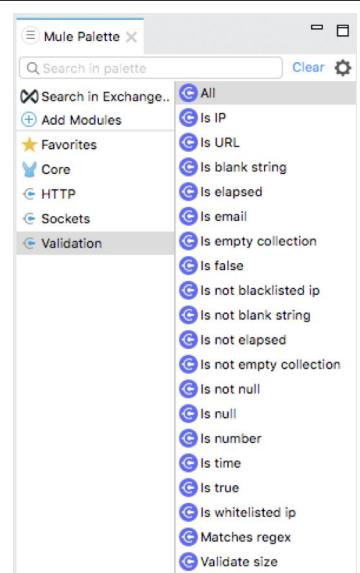


42

Validating conditions in a flow



- MuleSoft provides various **validation operations** to test (validate) if some conditions are true
 - And throw an error if the condition fails
- The condition is applied to a target DataWeave expression that usually involves the inbound Mule event
- Validation operations also provide an easy way to **throw some specific error type**



43

Modules that provide validation operations



- Generic validation operations
 - The Validation module
- Other modules provide validation operations for specific schema or data types
 - JSON - Validate schema
 - XML - Validate schema
 - APIkit - JSON validation, throw SOAP faults
 - Java - Validate type
 - HTTP Request operation - Validate response, identify success vs. failure status codes

Java - Validate type

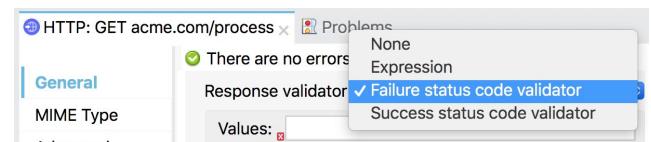
Display Name: Validate type

General

Class: com.mulesoft.training.flights.AmericanFlight

Instance: #[payload]

Accept subtypes: True (Default)



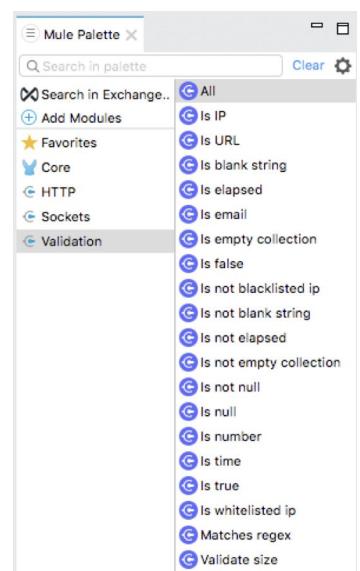
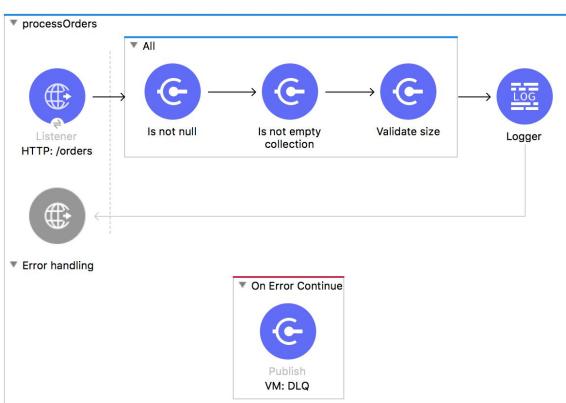
All contents © MuleSoft Inc.

44

Combining Validation module operations



- The Validation module provides generic validation operations
- The All scope can combine validation operations together



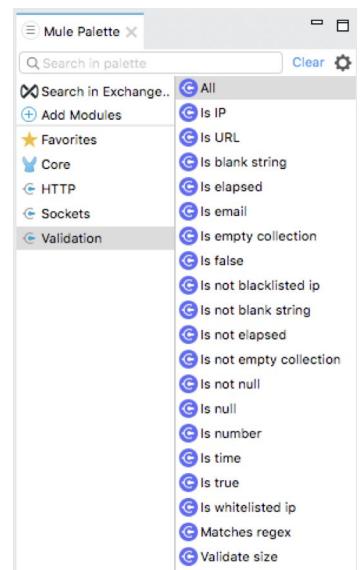
45

Using the Validation module without throwing an error



- Validation operations can be used directly inside DataWeave code
- In this case, the operation just returns true/false, it does not throw an error
- You can use this to write Choice router conditions

```
<choice>
  <when expression="#[ Validation::isEmail(vars.unknownVariable) ]" >
    <set-payload value="#[vars.unknownVariable ++ " is a valid email."]"/>
  </when>
  <otherwise >
    <set-payload value="#[vars.unknownVariable ++
      " is a not a valid email or a valid url."]"/>
  </otherwise>
</choice>
```

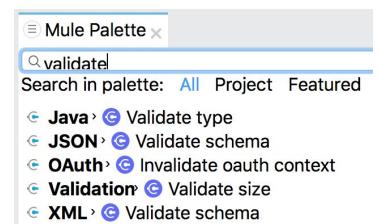


46

Validating schema in a flow



- There are also validation operations in other modules
 - XML, JSON, and Java**



47

- The Mule validation module, APIKit, and the other validation operations **throw an error** when the validation condition fails
 - It is **mandatory** to catch the thrown error to control the flow of events
 - This may also **degrade performance**
- As an architect, you may want to **avoid throwing** an error for every possible validation failure
 - Instead, design flows to **validate data using Choice routers or DataWeave** expressions to test and control the flow of events
 - This is like using if/else statements in a programming language like Java or C++

Reflection questions

- What are the tradeoffs of using a validation module vs. Choice routers vs. DataWeave code vs. custom code?
- How do validation modules relate to error handling components?
- What are the tradeoffs of throwing and handling an error in the same flow?
- How does APIkit handle invalid requests?
- What are the tradeoffs of using well defined schema in your flows to "fail fast"?
- Where specifically would well defined schema be helpful in flows, and what are the tradeoffs?
- How do these tradeoffs relate to decisions to use common data models?

Choosing message routing patterns for Mule applications



Mule events can be routed to different paths in a Mule application



- You already saw how a Choice router can change event processing in a flow based on **conditions** and flow control logic
- Sometimes processing must be completely **sequential** through the flow
 - Each event processor blocks the flow processing thread until it is finished
 - Mule flows can be configured to guarantee sequential event processing
- Other times you can run multiple event processing routes in **parallel**
- Mule applications have scopes and routers to support parallel processing

The issue with requiring sequential event processing in a Mule flow



- Context
 - Sometimes business logic requires incoming events to be processed **sequentially**
- Problem
 - Mule runtime is a **multithreaded** that allows processing of incoming events in parallel

How to configure sequential execution in a Mule flow



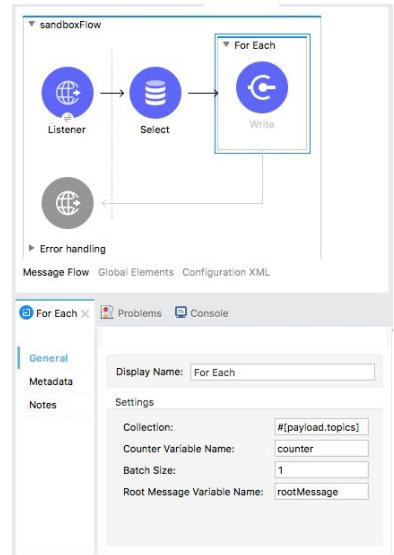
- Solution
 - Mule runtime engine allows to specify **max concurrency of 1** which ensures only one flow instance processes events at any point in time
 - The **For Each scope** (foreach) splits a payload into elements and processes them one by one through the components that you place in the scope
 - Akin to **for loops** in other programming languages
- Implementation

```
<flow maxConcurrency="1">
    <http:listener>
</flow>
```

Ways to implement sequential execution patterns in Mule applications



- By default For Each tries to split the payload
- For Each can split up iterable types
 - If the payload is a simple Java collection, the For Each scope can directly split the payload without any configuration
 - For non-Java collections, such as XML or JSON, you need to specify the iterable collection to use by setting the **collection** attribute
 - The value is evaluated from a DataWeave expression



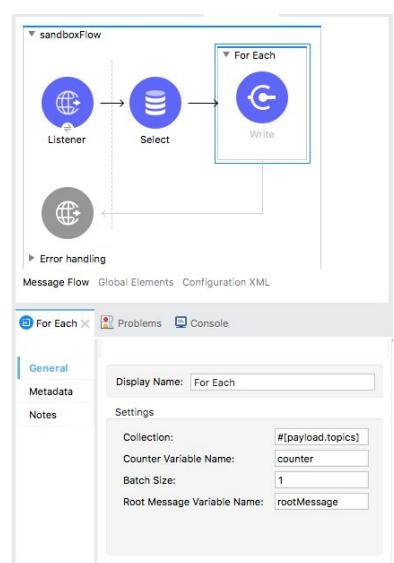
All contents © MuleSoft Inc.

54

Using a For Each scope's batch size to improve performance



- The **batch size** aggregates elements in the collection into smaller batch collections
 - Each smaller batch collection is passed together to the For Each scope's event processors
 - Instead of passing individual records one at a time
 - Not to be confused with the **Batch scope**
- A batch size of 1 means that individual elements (not 1-element collections) are passed on



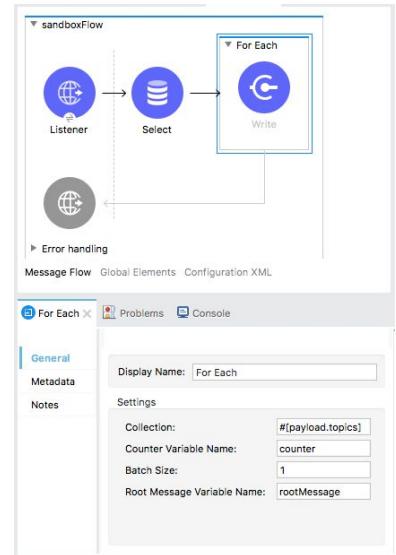
All contents © MuleSoft Inc.

55

Using the batch size to improve performance



- The default batch size is 1, so if a collection has 200 elements, the For Each scope iteratively processes 200 elements, each as a separate Mule message
- Using a batch size can reduce the number of iterations of the For Each scope
- Many connectors also support bulk operations, so can handle a collection of records all at once
 - This can avoid rate limits imposed by the backend service or endpoint



All contents © MuleSoft Inc.

56

Parallel processing



- Context
 - Sometimes business logic requires incoming events to be processed in **parallel**
- Problem
 - In a single flow, processors are **executed in sequence** and each processor execution is dependent on the execution of preceding processor
 - The processor that communicates with an external system might block (sequential) processing of the current Mule event by its flow
 - Such as requesting from a remote server or executing a lengthy database transaction

All contents © MuleSoft Inc.

57

Supporting parallel event processing in Mule flows

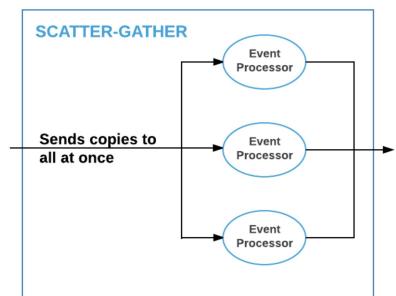


- To execute event processors concurrently and/or in parallel, the Mule runtime provides various processors
 - Scatter-Gather
 - Async scope
 - VM
 - JMS
- Parallel execution occurs using **different** threads from various thread pools
- If one event processor is **blocked**, other processors are able to **continue** processing without waiting for the completion of this processor

Implementing parallel processing using Scatter-Gather

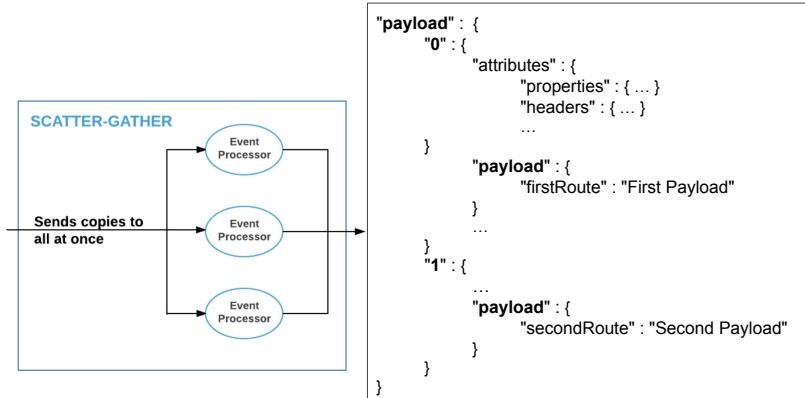


- The Scatter-Gather component is a routing event processor
 - Copies (scatters) the Mule event to multiple routes
 - Each route is processed in parallel
 - Parallel execution of routes may improve performance
- The Scatter-Gather completes after every route completes, or after a configured timeout expires
 - The default timeout is to wait forever
 - The timeout value is applied in parallel to each route
 - If a route's timeout expires, a MULE:TIMEOUT error is thrown for that route



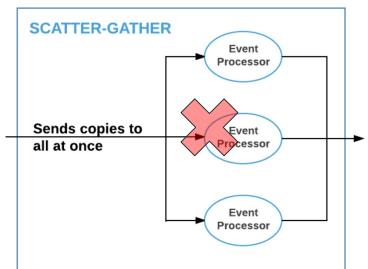
How the results from each route is gathered at the end of a Scatter-Gather

- If every route succeeds
 - After the last route finishes, each of Mule events are **gathered** into a single Mule event that is passed to the next event processor in the flow
 - The gathered Mule event payload contains each route's resulting Mule event
 - Access each route's result payload from the array #[payload..payload]



How errors from a Scatter-Gather are handled

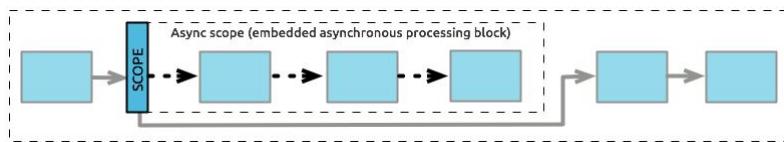
- If any route throws an error
 - An error of type MULE:COMPOSITE_ROUTING is thrown
 - The error object contains the result (Mule event or error) of every route, organized as successes or failures
 - The error object also includes the error from any routes that timed out
 - Event processing does not continue with the next event processor in the flow
 - Instead, the flows error handlers process the error as they would any other error type
- To recover from an individual route error, use a flow reference in each route and handle errors in the referenced flows



Implementing concurrent processing using an Async scope



- The event processors inside an Async scope execute concurrently with the event processors from the main flow
- Response** from the Async scope's event processing is **not accessible** nor is it returned to the main flow
- Errors inside the Async scope's event processors do not impact the main flow
 - And **do not use the outer flows error handler**
 - Use a Try scope inside the Async scope to customize error handling
 - Otherwise the default error handler is used



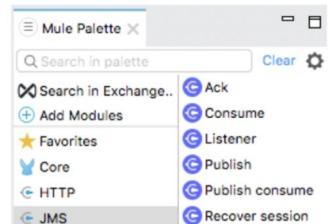
All contents © MuleSoft Inc.

62

Implementing concurrent processing using JMS destinations



- JMS connector provide async communication for intra and inter application through JMS destinations via a shared message broker
- JMS publish and consume/listener processors can be used to implement concurrent processing
- Supports various messaging models
 - Point-to-point queues
 - One-to-many topics



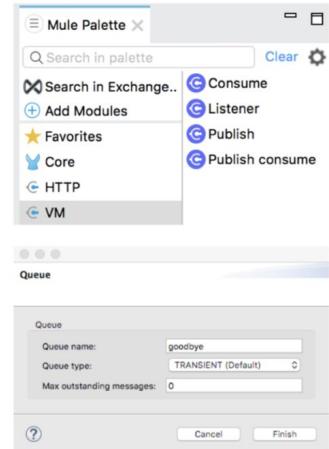
All contents © MuleSoft Inc.

63

Implementing concurrent processing using VM queues



- VM connector provides async communication for intra and inter application through asynchronous, inside-the-JVM queues
 - Define the VM connector in a Mule domain to share the VM queue between Mule applications
- The VM module's Publish, Consume, and Listener operations can be used to implement concurrent processing



All contents © MuleSoft Inc.

64

Implementing parallel out-of-order processing with a Batch Job



- A Batch Job processes a collection of records in a batch grouping at a time
- Each batch group is processed together through the Batch Job
- Each Batch Step processes in parallel in its own threads
- If a previous record blocks an earlier Batch Step, a later record can skip ahead to the next Batch Step and can complete the entire Batch Job out of order
- However, the Batch Job guarantees that each record moves through each Batch Step in sequential order

All contents © MuleSoft Inc.

65

- A Batch Step can include a Batch Aggregator scope
- The Batch Step can be configured with an aggregation count
- The Batch Step will collect processed records until the aggregation count is reached
- Then the aggregated records are sent together to the event processors inside the Batch Aggregator scope
- Some connectors support bulk operations that can directly handle the aggregated payload
 - For example, the Salesforce and Database connectors
 - This can increase throughput at the expense of timeliness

Reflection questions

- What are the tradeoffs to process collections of elements/records with a For Each scope vs. a Batch Job vs. all at once with a single bulk operation?
- What are the tradeoffs of using bulk operations inside a Batch Job vs. all at once?
- What are the tradeoffs of using a single bulk operation vs. processing each record in a For Each scope?

- Is parallel processing (such as in a Scatter-Gather) always faster?
- What external factors might limit parallel processing?
- What other tradeoffs need to be considered to decide between sync, async, and parallel processing?
- Which use cases are well suited to bulk operations, and what external factors might affect this decision?

Applying message transformation, validation, and routing patterns



Applying message transformation, validation, and routing patterns to use cases

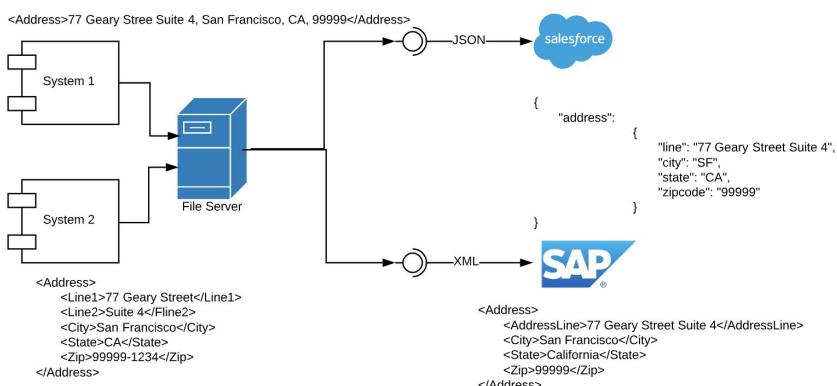


- Now you can apply the design patterns from this module to some of the course use cases
- This includes deciding when and how to apply design patterns such as
 - The best way to transform data between endpoints
 - When and how to design common data models and data mappings
 - The best way to write defensive flows and the best routing patterns to use
 - The best way to handle errors

Exercise 5-1: Apply message transformation and routing patterns to a systems integration use case

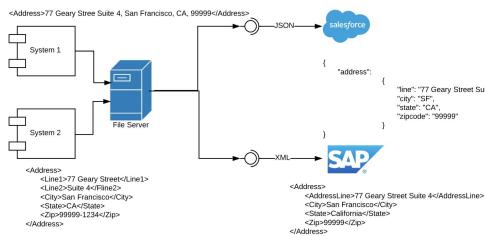


- Design integration between multiple inbound and outbound systems
- Apply enterprise integration design patterns efficiently and effectively to meet requirements of a specific scenario



Exercise context

- Enterprise integration requirements
 - Two systems **System 1** and **System 2** send files with the same kind of information to a **shared file server**
 - System 1 and System 2 both **write XML files**, but each system uses a different data model to represent an **Address** object
 - Data from both systems must also be **sent** to **SAP** and **Salesforce** systems
 - Which again use **different data models** for **Address** objects



Exercise steps

- Decide which data transformation processes are required or not
 - From System 1 to Salesforce
 - From System 1 to SAP
 - From System 2 to Salesforce
 - From System 2 to SAP
- Decide ways to reduce multiple transformations
 - Can a standardized CDM across the systems reduce the number of transformations?
 - How might a CDM decouple future schema changes in System 1 or System 2 or the SAP and Salesforce systems?
 - How might a CDM help if a new System 3 needs to be connected?
- Design Mule integration flow(s) that use a CDM for this scenario

Exercise solution: Without a CDM



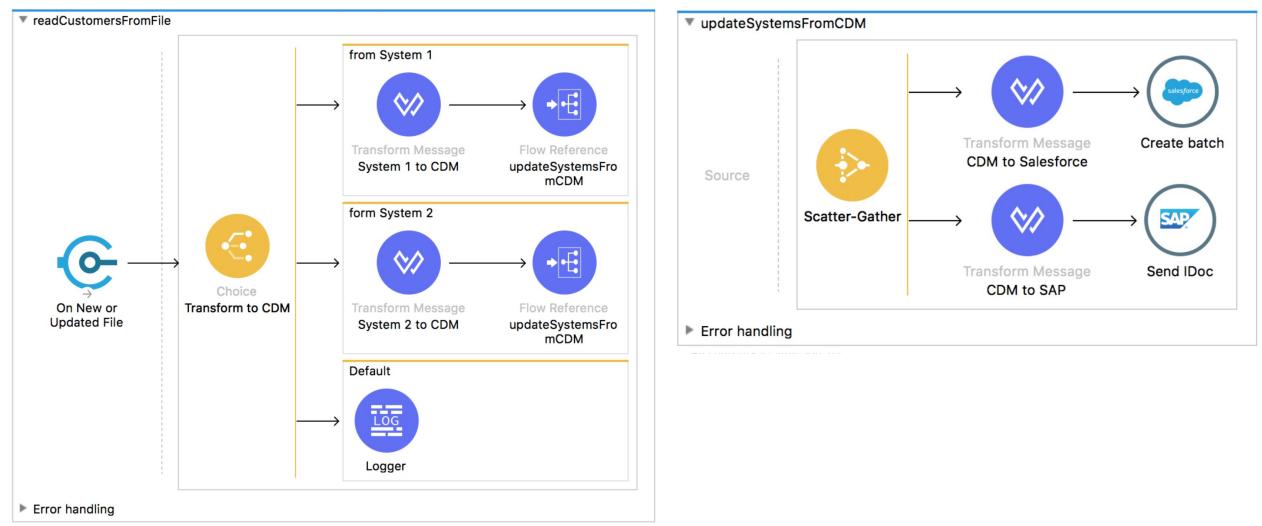
- Have to create different transformations for 2x2 routes
- Have to create separate transformations to Salesforce and SAP twice
 - These mappings tend to be complicated and vendor specific

80



Exercise solution: With a CDM

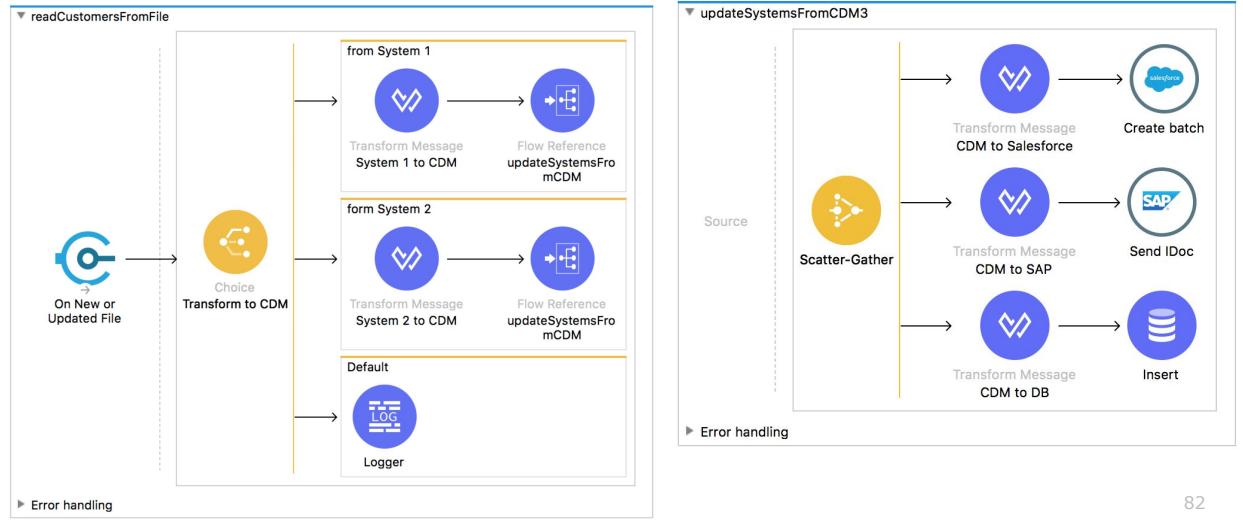
- Transformation to Salesforce and SAP are reused



Exercise solution: Adding a third output endpoint



- Adding a third endpoint is now decoupled from the CDM step



82

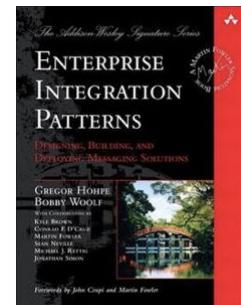
Summary



- DataWeave can simplify translating between common input and output data types
- Mule events often need to be validated and routed within a flow
- Defensive code is implemented with a combination of choice routers, validator components and DataWeave expressions, and error handling
- Scatter-Gather, Async scopes, and Message queues can help process events in parallel

References

- Hohpe & Woolf, 2003
 - Describes 65 patterns for EAI and MoM
 - <http://www.eapatterns.com/>
 - Mule runtime has implemented many of these patterns





Module 6: Designing Mule Application Testing Strategies



Goal



The screenshot displays the Anypoint Studio interface with the following panels:

- Package Explorer:** Shows the project structure with files like `order-processing`, `src/main/mule`, `src/main/java`, `src/main/resources`, `src/test/java`, and `src/test/munit`. The file `order-processing-test-suite.xml` is selected.
- order-processing-test-suite:** A test configuration window for the selected suite. It includes sections for **Behavior** (with a note to "Add mocks and spies here"), **Execution** (with a "Flow Reference" icon and "Flow-ref to processOrder" text), and **Validation** (with an "Assert that" icon). Buttons for "Message Flow", "Global Elements", and "Configuration XML" are at the bottom.
- MUnit:** A test results window showing "Run: 1/1", "Errors: 0", and "Failures: 0". It also includes a "Sort by" dropdown and a list of tests: `order-processing-test-suite.xml` and `order-processing-test-suite-processOrderTest`.
- order-processing:** A message flow editor window showing a flow from an "HTTP /order" listener to a "Transform Message" processor. Buttons for "Message Flow", "Global Elements", and "Configuration XML" are at the bottom.
- Coverage:** A coverage report window with tabs for "Errors" and "Coverage". It has a checkbox for "Check covered message processors", a "Generate Report" button, and a summary: "Overall coverage: 100.00%". Below this, it shows coverage details for `orderprocessing.xml` and the `processOrder(100.00%)` message processor.

All contents © MuleSoft Inc.

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



- Understand why testing is important in an integration project
- Identify the MuleSoft provided testing framework and tools for testing during the software development lifecycle
- Design testing strategies for a Mule application
- Define test coverage for flows in a Mule application

Designing testing strategies for Mule applications



- To **find problems early** in the **development cycle** and **fix them quickly**
- As executable documentation of the code's behaviour
- To Minimize risk for developers to later make changes
- To make collaboration between developers easier



- To find problems quickly **every time** code is changed
- As a necessary step in continuous deployment, continuous integration, and continuous delivery pipelines



The architect's responsibilities related to designing testing strategies



- It is important to build a comprehensive **test suite** of relevant tests for each Mule project
 - To ensure **consistent, predictable, and reliable deployments** and runtime operations
- As an architect, you are responsible for defining these testing strategies, particularly as they relate to Mule applications

Types of testing that are particularly relevant to Mule applications



- Unit testing
 - Verifies the **functionality of a specific unit** (often a flow or subflow)
 - If the flow exposes an interface, request data and any external calls or dependencies are usually mocked or stubbed-out
- Integration testing
 - Verifies **all interactions to/from interacting systems** are **functional**
 - Test the actual requests and/or responses exchanged with external systems
- Performance testing
 - Measure, validate, or verify performance characteristics of the system, such as **scalability, reliability, and resource usage** under **various workload**

Type of tests **not** particular to a Mule application



- Other types of testing that relate more to the general lifecycle of any Java application may not need specific Mule testing tools
 - Such as destructive, resilience, functional, and regression testing
 - These testing strategies may already be in place within the organization for every application deployment

Reflection questions: Unit testing



- What are some examples where unit testing is helpful, and how would you implement unit tests?
- What are the general features of a unit test of
 - An HTTP Request?
 - A database select vs. a database insert, update, or delete operation?
 - A DataWeave transformation?
 - A Scatter-Gather component?
 - A For Each scope and its components?
 - Custom Java code?

- You'll learn more about performance testing later in the course, but for now think about these questions
 - What network considerations might affect performance testing?
 - What choice of tools might affect performance testing?
 - What are the tradeoffs of meeting various performance goals, and what other goals might conflict with performance goals?

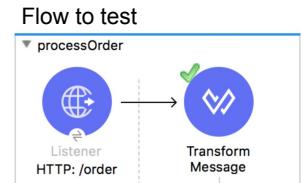
Testing Mule applications using MUnit



How Mule flows can be tested



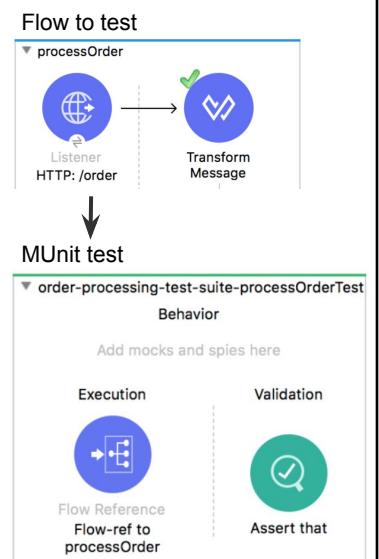
- **JUnit** is often used to build unit tests specific functionality of Java code
 - Usually to test one **method** call, or a Java class's constructor
 - External dependencies are replaced, mocked, or stubbed
 - Input data is provided
 - Assertions about the result are tested and recorded
- In a Mule application, a flow is logically equivalent to a Java method



How Mule flows can be tested using MUnit



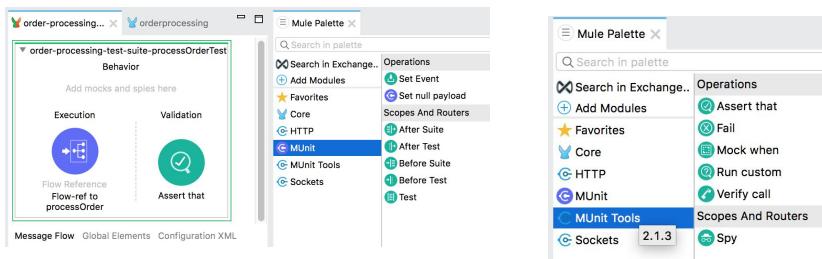
- Anypoint Studio includes **MUnit** to
 - To build **test suites** focused specifically on how flows in a Mule application behave
 - **Automatically** generates testing stub flows for flows
 - The developer then fills in the testing **assertions** (conditions and logic) using Mule event processors from the MUnit modules



Using MUnit to test Mule applications



- **MUnit** is a test suite specifically targeted to testing Mule applications inside a Mule runtime
 - Builds test suites focused on Mule application details
 - Each test is itself a Mule flow, using components from the **MUnit** and **MUnit Tools** modules
 - The testing Mule flows execute in a Mule runtime and report results



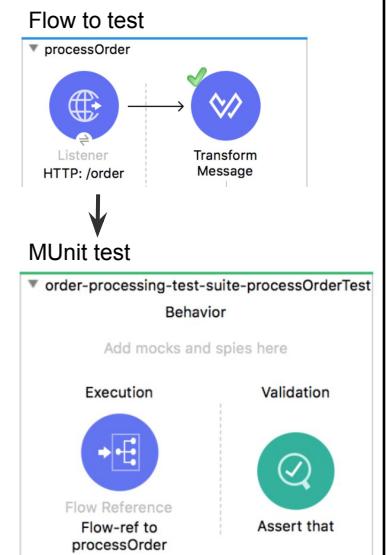
All contents © MuleSoft Inc.

15

Using MUnit to build Mule application testing suites



- MUnit allows you to
 - Design and create **whitebox** test cases as Mule flows
 - Tests the internal workings and behaviors of Mule flows
 - Each test is itself a Mule flow
 - Run tests as Mule flows in a Mule runtime
 - **Manually** in Anypoint Studio
 - **Automatically** as part of Maven based **CI/CD** build process
 - View test results and coverage inside Anypoint Studio



All contents © MuleSoft Inc.

16

MUnit tests are also Mule flows



- The MUnit module provides event scopes and event processors to test flows and event processors within flows

```
<munit:config name="new-test-suite.xml" />
<munit:test name="new-test-suite-ordersUnitTest" description="Test" tags="sales, unit">
    <munit:execution>
        <flow-ref doc:name="Flow-ref to orders" name="orders"/>
    </munit:execution>
    <munit:validation>
        <munit-tools:assert-that doc:name="Assert that payload is not null"
            expression="#{payload}" is="#{MunitTools::notNullValue()}" message="Payload is not null"/>
    </munit:validation>
</munit:test>
<munit:test name="new-test-suite-accountBalanceTest" description="Test" >
    <munit:execution>
        <flow-ref doc:name="Flow-ref to accountBalance" name="accountBalance"/>
    </munit:execution>
    <munit:validation>
        <munit-tools:assert-that doc:name="Assert that" expression="#{payload}"
            is="#{MunitTools::notNullValue()}" message="Payload is not null"/>
    </munit:validation>
</munit:test>
```

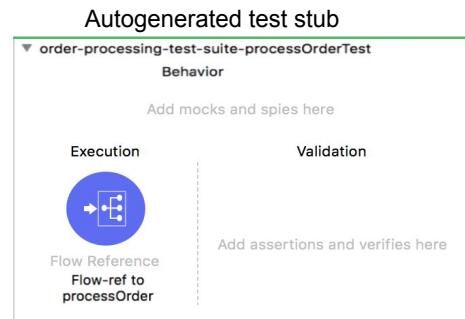
All contents © MuleSoft Inc.

17

The structure of autogenerated MUnit tests



- Each MUnit test case created by Anypoint Studio has
 - An Execution section with a **configured event source**
 - An **empty** Validation section
- Execution section usually calls a flow in the Mule application
 - This bypasses the flow's Event Source



All contents © MuleSoft Inc.

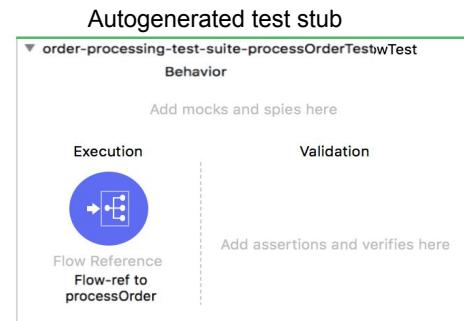
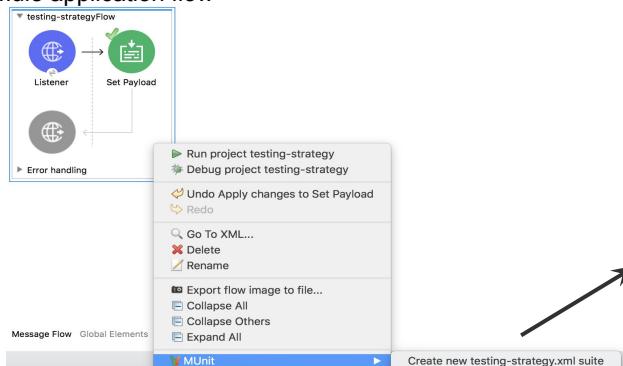
18

Autogenerating MUnit test suites using Anypoint Studio



- Anypoint Studio can create stubs MUnit test suites from a Mule application's flows
 - The test suite is a new Mule XML configuration file with stub flows
 - This way the testing flows do not clutter the actual Mule app XML files

Mule application flow

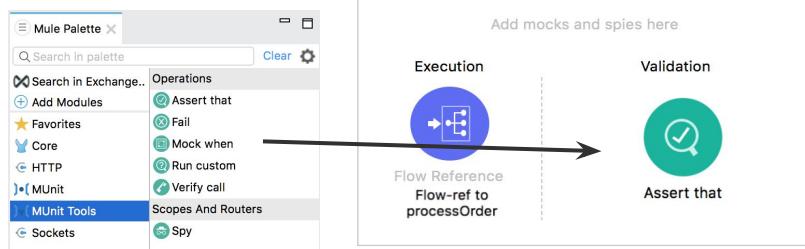


19

Design and create test cases



- Validation should be added to each test case
 - To add the logic to compare the expected result vs. the actual result of the test case

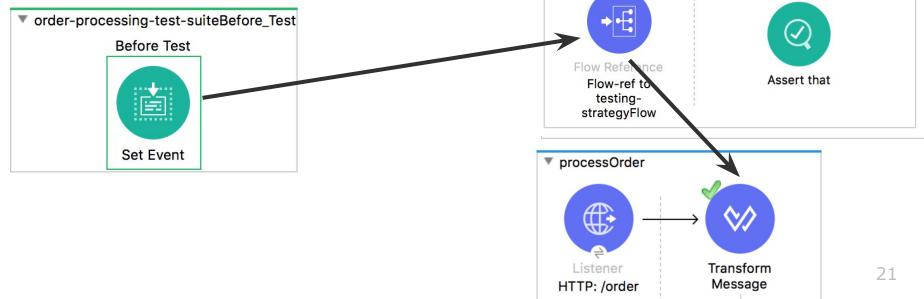


20

Passing known event data into an MUnit test



- A flow reference in the Execution section bypasses the referenced flow's Event Source
- A **Before Test** scope can set the Mule event passed to the Execution scope
 - Allows the test to input a known event with expected results from each flow component



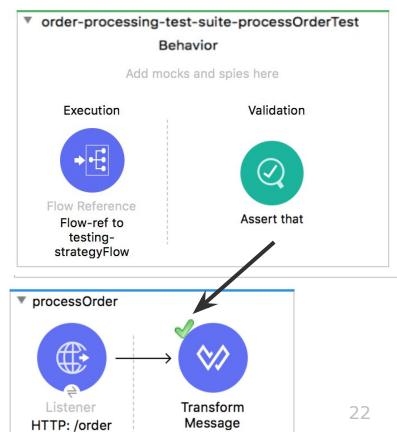
All contents © MuleSoft Inc.

21

Viewing test results and coverage in Anypoint Studio



- When a flow passes tests, the flow components display a green check mark
- This can also visually show you how much of a Mule application is covered by the test suite or test suites



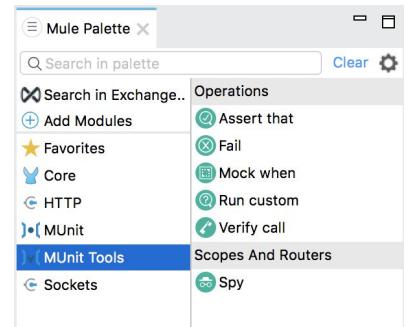
All contents © MuleSoft Inc.

22

Test components used to build MUnit tests



- **Mock** any processor
 - **Deterministically return expected data or throw a certain error** for certain event processors
 - Focuses the unit test on the particular flow
 - Ignores dependencies outside the flow, such as results of a connector, flow ref, DataWeave transformation, variables, and attributes
- Define **assertions**
 - Set boolean conditions to compare expected vs. actual data at a specific point in a flow
- Spy on flow components
 - A scope to **validate (assert) the Mule event state** before and after a flow component



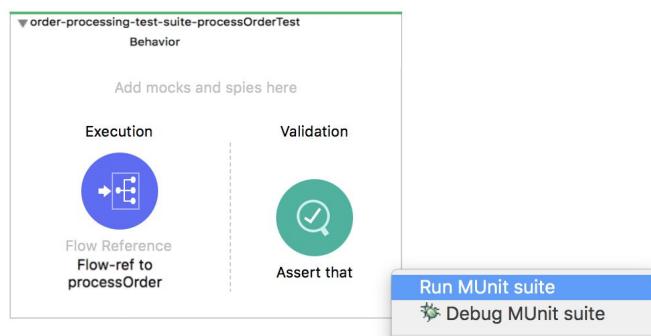
All contents © MuleSoft Inc.

23

Running MUnit test suites



- MUnit test cases are deployed and run in a dedicated Mule runtime
 - The Mule runtime is started for the tests by Anypoint Studio or by the MUnit maven plugin
 - After the selected test suite(s) run, this Mule runtime terminates



All contents © MuleSoft Inc.

24

- MUnit test suites can also be run automatically outside of Anypoint Studio
- MUnit is fully integrated with Maven through a Mule Maven plugin
 - Libraries are available in the MuleSoft public nexus
 - MUnit test suites are defined as elements inside a Mule application XML
 - Are triggered during a Maven test phase
 - Works with surefire-reports
- This allows tests to run
 - Automatically as part of any Maven build
 - Such as locally at the developer's machine
 - Whenever code is changed and checked back in to an online repository, or other CI/CD pipeline

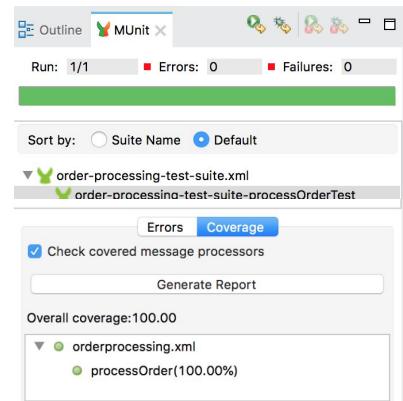
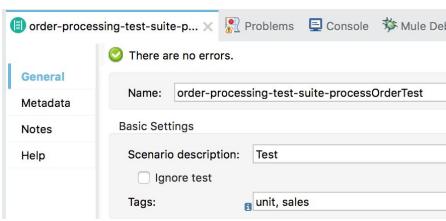
How to mock data to support MUnit tests

- You already saw that MUnit can **mock** any event processor and replace the output with mocked data
 - Including connectors, flow references, DataWeave, variables, and attributes
- There are also **other ways to mock data** going through a Mule flow
 - API Manager provides mocking services for APIs
 - Other third party products, servers, and online sites are commonly used to mock external responses for other types of external systems

MUnit features to select tests to run and view test results



- Code coverage reports
 - Reports the percent of code that was tested, visually in Anypoint Studio and in generated reports
- Test tags
 - Each MUnit test can include tags
 - Can run only tests matching selected tag(s)
 - For example to tag unit vs. integration tests



All contents © MuleSoft Inc.

27

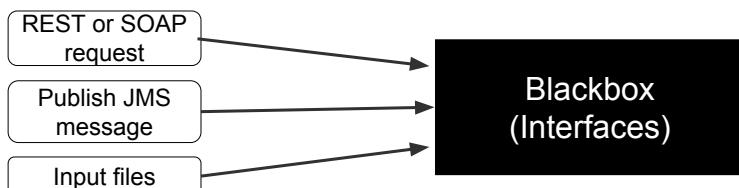
Designing integration and performance tests of Mule applications



- MUnit is also a whitebox integration testing framework for Mule applications
 - Assertions
 - Code coverage reports (generated reports)
 - Test tags

Features of blackbox (functional) integration testing

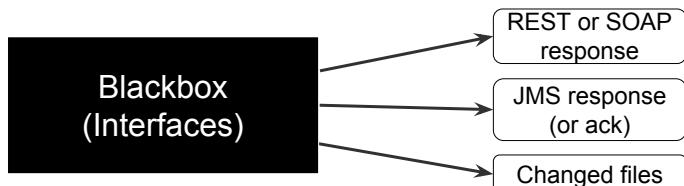
- Interact with the component-under-test only through its published interface
 - Invoke REST APIs or SOAP web services exposed by the component
 - Send JMS or other messages to a queue or topic on which the component listens
 - Create files in a folder which the component watches
- Have no knowledge of implementation details of the component



Other features of blackbox (functional) integration testing

MuleSoft

- Register the component's response or observable behaviour (side effects)
 - REST or SOAP response
 - Resulting (response or follow-up) JMS messages
 - Disappearance or renaming of consumed files or creation of output files
- Assert that the response/behaviour from/of the component adheres to its published specification
 - RAML definition or WSDL document
 - Human-readable documentation and functional specification



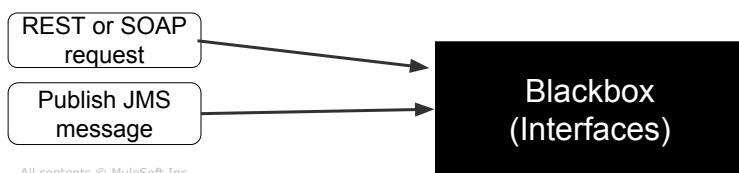
All contents © MuleSoft Inc.

32

Minimum support needed for Blackbox (functional) integration **testing tools** to generate requests

MuleSoft

- Ability to trigger the component-under-test through its published interface
 - Send HTTP request: plain, REST, SOAP
 - Send JMS messages



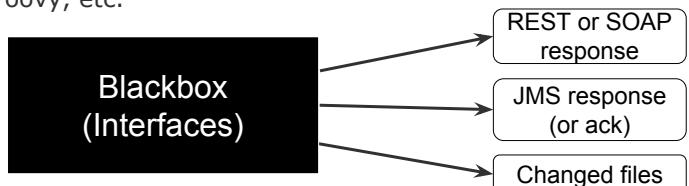
All contents © MuleSoft Inc.

33

Minimum support needed for Blackbox (functional) integration **testing tools** to **test responses**

MuleSoft

- Ability to record the component's response
 - Receive HTTP responses and JMS messages
- Ability to define assertions that validate the response
 - Validate XML and JSON docs for well-formedness and against a schema
 - Validate HTTP response status codes (200 - 599)
 - Expressions on HTTP or JMS header values, HTTP response body, JMS payload, parts thereof, etc.
 - Requires an expression language or full-fledged programming language, such as JSON path, XPath, Groovy, etc.



All contents © MuleSoft Inc.

34

Blackbox integration testing should be able to be automated

MuleSoft

- Should be able to be launched both
 - Manually and interactively
 - Automatically from Continuous Delivery (CI/CD) pipelines
- Tools should support Maven plugins or Jenkins pipeline plugins



35

Popular tools for blackbox (functional) integration testing



- These tools fulfill most of the previously listed requirements:
 - SOAPUI
 - Open-source and commercial
 - Restlet Client
 - No explicit support for SOAP over plain HTTP and no JMS support
 - REST-assured
 - Integrates with JUnit, so tests are written in Java, Scala or Kotlin
 - No explicit support for SOAP over plain HTTP and no JMS support
 - But any JVM library usable within JUnit tests

Performance Testing



- MuleSoft does not provide particular performance testing tools
- As an architect, to plan for performance testing earlier in development is mandatory for application where performance is KPI
 - JMeter or BlazeMeter or any open source performance testing can be used for performance testing
- Later in the class, the **Optimizing Performance** module has detailed discussion about performance testing and how to measure the performance of a Mule application

- The **MUnit Coverage** feature provides metrics on what percentage of a Mule application's code has been tested by a set of MUnit tests
 - **Flow coverage metrics**
 - Reports how many flows, subflows, and batch jobs were covered
 - **Resource coverage metrics**
 - Reports on each Mule configuration file under src/main/mule
 - **Application overall coverage metrics**
 - An average of the flow and resource coverage metrics
- For reliable CI/CD pipelines, aim for MUnit test coverage of 80% or above

Exercise 6-1: Define a testing strategy for a use case

- Define the types of testing required for a use case
- Document a testing strategy for a use case
- Document the code coverage for test cases

Exercise steps



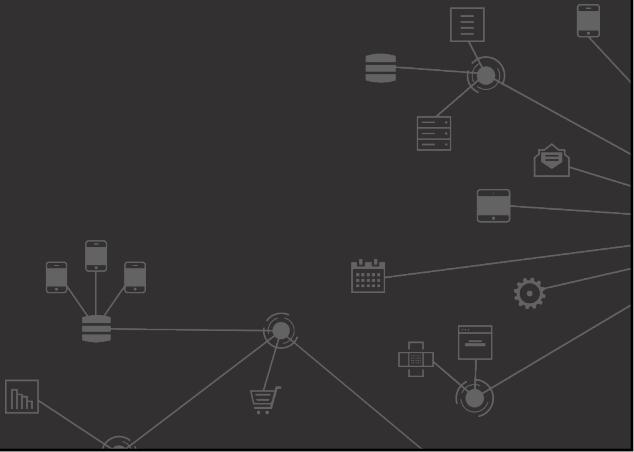
- Reread the use case and identify the testing requirement for the class use cases
- Define the type(s) of testing required to support the uses cases
- Document the code coverage by an MUnit test suite
 - Note: Due to the time limits of this class, do not document individual test cases

Exercise solution



- Open the exercise solutions file
- Compare your documented testing strategy with the exercise solution's testing strategy

Summary



Summary



- It is important to **identify, define, and implement automated tests early in a project** to help developers to find bugs early in the development lifecycle
- Good testing strategies give **developers confidence** about changing code throughout the entire lifecycle of a Mule application
- With MUnit, unit testing and whitebox integration testing is easily **automated and incorporated with CI/CD** and helps to improve code quality
- Documenting test coverage for application is another important part of testing strategy
- Performance testing is not always required in every Mule application

- What are some examples where unit testing is helpful, and how would you implement unit tests?
- How are integration tests of a flow different from unit tests?
- How is source and target data retrieved vs. mocked for these types of tests?
- What factors affect or limit the effectiveness of an integration test?
- What types of performance tests are common for Mule flows?
- Should testing use average load, worst case load, or some other model?



Part 2: Operationalizing Integration Solutions



Goal

The MuleSoft logo, featuring a blue stylized 'M' inside a circle followed by the word 'MuleSoft' in a blue sans-serif font.

A circular diagram representing the API lifecycle. It consists of three concentric rings. The outermost ring is dark blue and contains the words 'Analyze', 'Monitor', 'Deploy & Register', 'Secure', 'Version', 'Service with APIs', 'Respond', 'Scale', and 'Troubleshoot'. The middle ring is light blue and contains 'Build', 'Test', 'Validate', 'Simulate', 'Design', and 'Feedback'. The innermost ring is white and contains 'API Spec (RAML)'. Arrows indicate a clockwise flow between the stages.

A screenshot of the 'Runtime Manager' interface from the MuleSoft Anypoint Platform. It shows a dashboard with various metrics and status indicators for runtime environments.

A screenshot of the 'API Analytics' interface from the MuleSoft Anypoint Platform. It displays a world map with data overlays and a bar chart showing requests by application and platform.

A screenshot of the 'API Manager' interface from the MuleSoft Anypoint Platform. It shows a list of APIs with columns for name, description, version, and status.

A blue cloud icon containing a white MuleSoft logo, representing 'MuleSoft-hosted infrastructure'.

MuleSoft-hosted infrastructure

A blue cloud icon containing a white MuleSoft logo, representing 'Customer-hosted in the cloud'.

Customer-hosted in the cloud

A blue server icon containing a white MuleSoft logo, representing 'Customer-hosted on-premises'.

Customer-hosted on-premises

All contents © MuleSoft Inc.

2

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

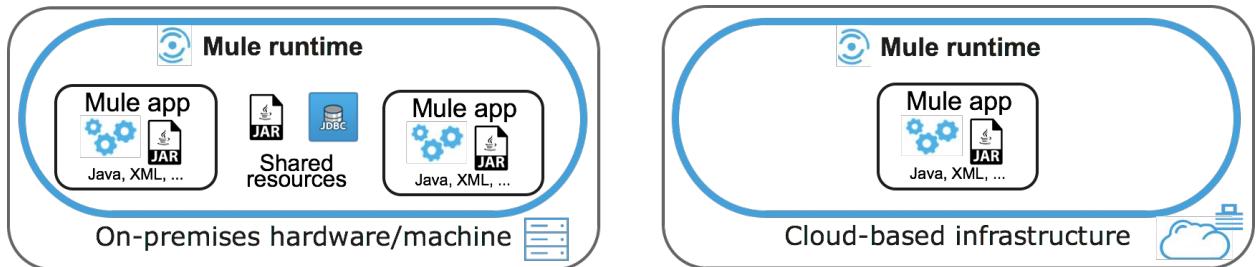


- Decide and develop appropriate and optimal deployment strategies
- Decide the best ways to preserve and manage Mule application state
- Design effective and sufficient logging and monitoring strategies
- Create an efficient and automated software development lifecycle (SDLC)



Module 7: Deciding and Developing a Deployment Strategy





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

- Identify the service and deployment models supported by Anypoint Platform
- Decide deployment options for various scenarios
- Design containerized deployments for Mule runtimes

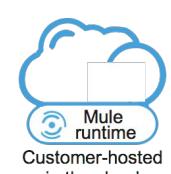
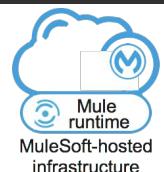
Distinguishing between various Anypoint Platform service models



Identifying the various Mule runtime hosting service models provided by Anypoint Platform



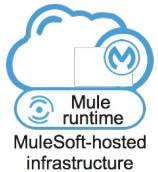
- The **MuleSoft-hosted** runtime plane uses cloud-based infrastructure (iPaaS)
 - **Provisioned** and managed by MuleSoft
 - The service is called **CloudHub**
- The **Customer-hosted** runtime plane is entirely provisioned and managed by the **customer** in
 - Non-MuleSoft-hosted cloud environments
 - Examples: AWS, Azure, Google Cloud, PCF
 - On-premises infrastructure, on bare-metal, virtual machines, or containers like docker



Features of the MuleSoft-hosted runtime plane



- Deployment to the **MuleSoft-hosted** runtime plane uses cloud-based infrastructure (iPaaS)
 - A new virtual machine is automatically provisioned for the new Mule runtime
 - This is called a **CloudHub worker**
 - Each Mule application is deployed to a separate Mule runtime which runs in a completely isolated CloudHub worker
- A Mule application can be scaled vertically or horizontally
 - Horizontal scaling: The Mule application can be automatically deployed to multiple CloudHub workers
 - Vertical scaling: A Mule application can have its CloudHub worker resized to a larger or smaller vCore size



Other components of a MuleSoft-hosted runtime plane



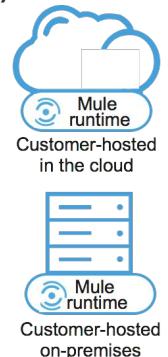
- MuleSoft provides a **load balancing** service
 - Distributes inbound HTTP requests among multiple CloudHub workers
 - Provides automatic redirection to new CloudHub worker(s) after a Mule application is resized or restarted
 - Resulting in **zero downtime** for the Mule application
 - A shared VPC restricts access to CloudHub workers to specific ports
 - 80/443 and 8081/8082
- MuleSoft implements a distributed **object store** service
 - Object stores and VM queues are implemented using this object store service
 - These can be configured so data survives restarting the Mule application



Ways to deploy to multiple Mule runtimes in customer-hosted infrastructure



- With just the Mule runtime software, the customer is responsible for coordinating how a Mule application is deployed to multiple Mule runtimes, and manage any load balancing or other scaling
- Runtime Manager Fabric** is additional software provided by MuleSoft to scale out customer-hosted Mule runtimes
 - Mule applications deploy to isolated Mule runtime which run in a docker container
 - The containers can be deployed to any cloud service such as Google Cloud, MicroSoft Azure, and AWS
 - Provides additional services and infrastructure to provide CloudHub like services that run on-premises
 - Like zero-downtime redeployment and load balancing



Comparing Anypoint Platform control plane options

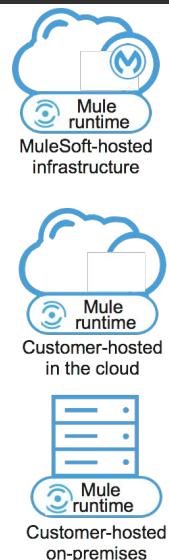


Plane type	MuleSoft-hosted Anypoint Platform	Anypoint Platform - Private Cloud Edition
Managed by	<ul style="list-style-type: none">MuleSoft	<ul style="list-style-type: none">Customer
Features	<ul style="list-style-type: none">Full Anypoint Platform MuleSoft-hosted control planeCan deploy Mule apps to CH workersView logs in Runtime Manager for Mule apps deployed to CloudHub workers	<ul style="list-style-type: none">Most Anypoint Platform featuresCan not deploy Mule apps to CH workersLogs are not available in Runtime Manager
SLAs	<ul style="list-style-type: none">MuleSoft automatically patches and updates	<ul style="list-style-type: none">Customer must install patches and updates

Identifying the various Mule runtime hosting service models provided by Anypoint Platform



- In each service model, the **same Mule runtime binary executable** is used in a particular runtime plane (infrastructure)



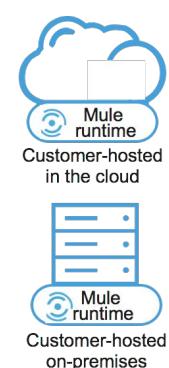
All contents © MuleSoft Inc.

13

Customer responsibility when using Mule runtimes installed in a customer-hosted runtime plane



- For customer-hosted, the customer is responsible for provisioning and managing all
 - Hardware, virtual machines, cloud environments, and operating systems
 - Networks, proxies, load balancing, and high availability services
 - Java versions, security enhancements



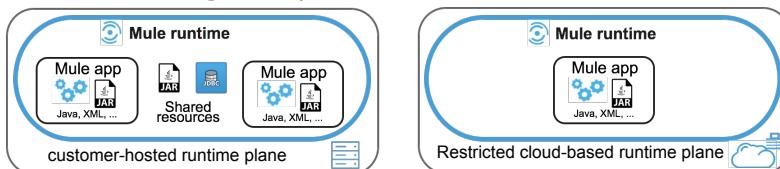
All contents © MuleSoft Inc.

14

The service models for customer-hosted runtime planes



- The **Customer-hosted** runtime plane (either on bare metal or in virtual machines) can be configured to
 - Allow deployment of multiple Mule applications and domain to the same Mule runtime
 - Host multiple Mule runtimes
- Like CloudHub, other **cloud-based** infrastructure (like PCF) can also be locked down
 - May not allow deployment of multiple Mule applications or Mule domains
 - May not allow hosting multiple Mule runtimes in the same virtual machine



All contents © MuleSoft Inc.

14

15

MuleSoft-hosted vs. Customer-hosted runtime plane



MuleSoft-hosted



Customer-hosted



Each Mule runtime is automatically installed in a separate CloudHub worker

Mule runtimes can be deployed to various types (on-premises or cloud) of customer-hosted infrastructure

Can host one Mule application in a dedicated CloudHub worker

Some customer-hosted infrastructure can run multiple Mule applications in one Mule runtime

No contention for host resources

Can have contention for host resources

Automatic upgrades/patches

Customer must schedule and install updates

Globally available

Customer must provision across regions and multiple premises

MuleSoft provided security and SLAs

Customer defines and controls security and SLAs

Managed by the MuleSoft-hosted Anypoint Platform control plane

Managed by the MuleSoft-hosted or customer-hosted Anypoint Platform control plane

All contents © MuleSoft Inc.

16

Reflection questions: Runtime planes



- What components make up the MuleSoft-hosted runtime planes?
- What components make up various customer-hosted runtime planes?
- What use cases would prohibit using a MuleSoft-hosted runtime plan
- What are the tradeoffs of these runtime plane options?
- What use cases would benefit from a hybrid collection of runtime planes?
- What are the tradeoffs in using a hybrid model?
- How could a hybrid model help an enterprise move to the Cloud?

Reflection questions: Runtime planes



- For what use cases is the MuleSoft-hosted runtime plane the best option?
- For what use cases is a customer-hosted control plane the best option?

- What is the difference between a control plane and a runtime plane?
- What are the components of the MuleSoft-hosted control plane?
- What use cases would prohibit using a MuleSoft-hosted control plane?
- What use cases would use a combination of control planes, and what would be the tradeoffs and benefits?
- For what use cases is the MuleSoft-hosted control plane the best option?
- For what use cases is a customer-hosted control plane the best option?

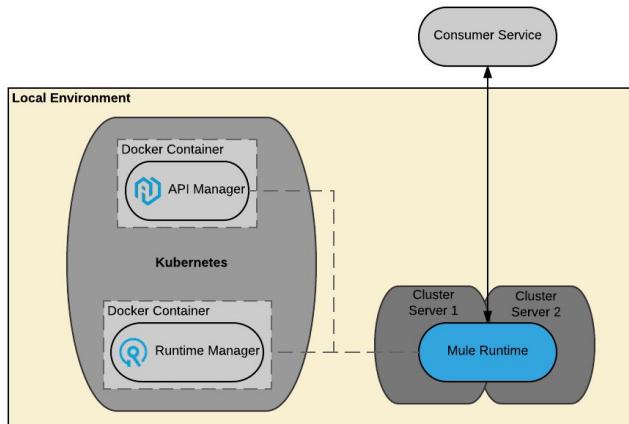
Distinguishing between various Anypoint Platform deployment models



Deployment models for Anypoint Platform - Private cloud edition



- For customers with **strict regulatory or compliance requirements**

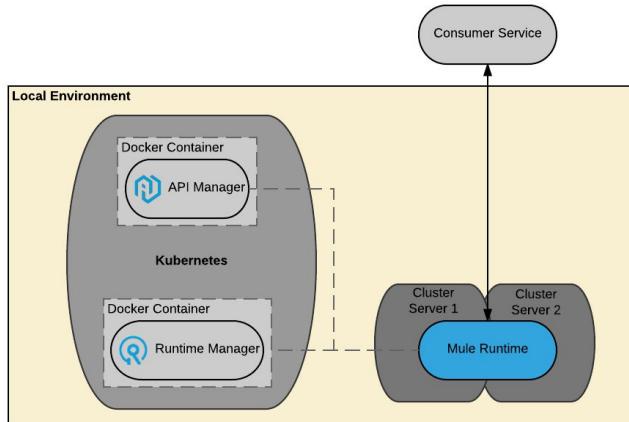


21

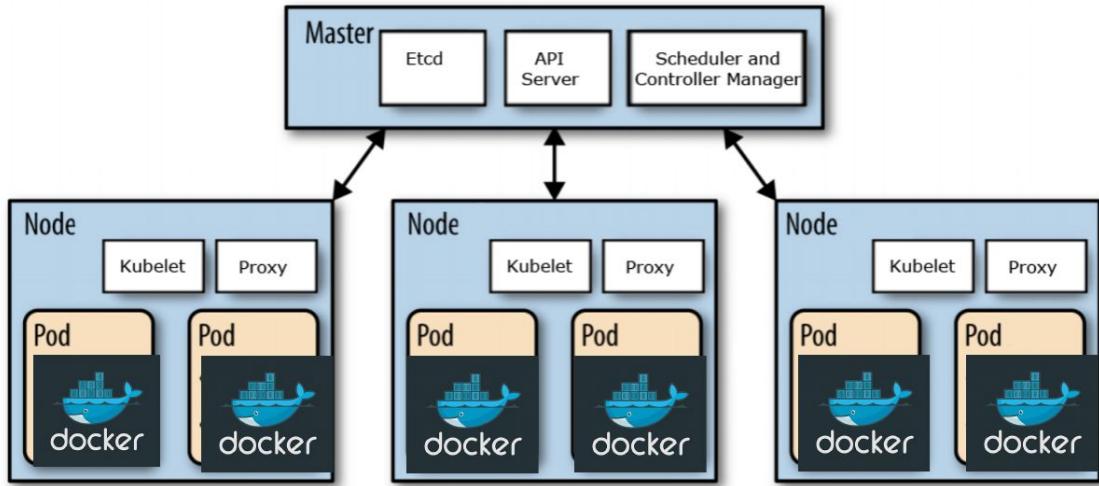
Deployment models for Anypoint Platform - Pivotal cloud foundry (PCF)



- Same as Anypoint Platform - Private Cloud Edition, except the runtime plane is deployed on PCF instance
- Like **CloudHub**, **one Mule application per Mule runtime**

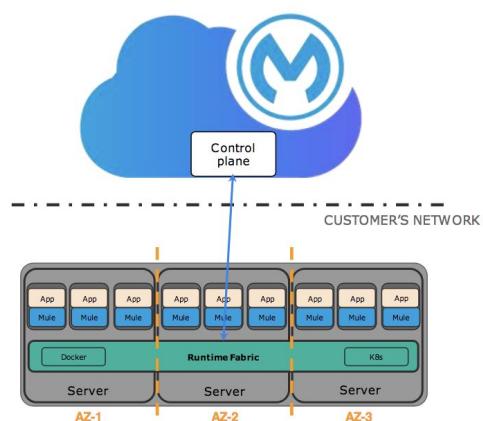


22

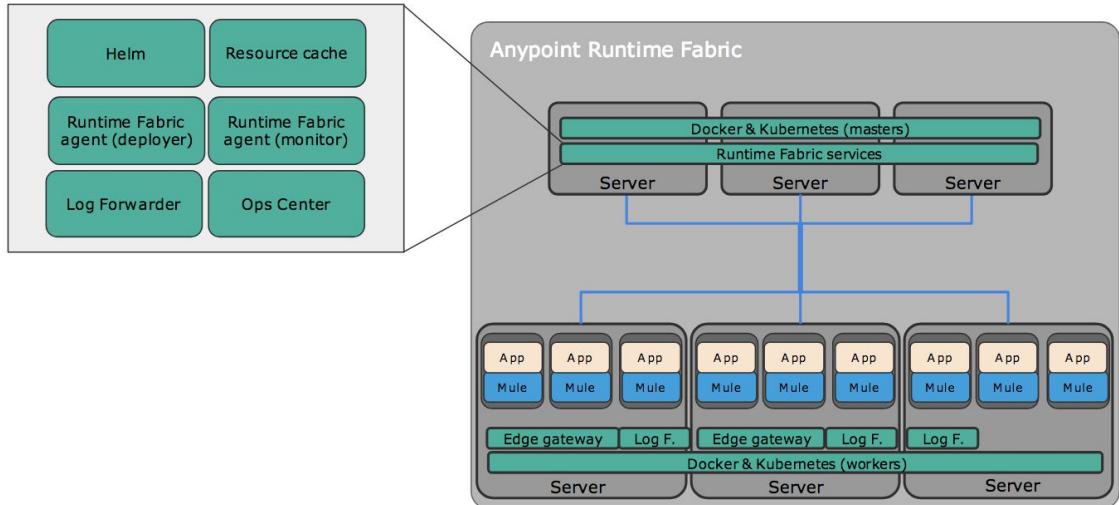


Containerized deployment on Anypoint Platform- Anypoint Runtime Fabric

- Anypoint Runtime Fabric orchestrates and automates the deployment of Mule runtimes into containers in any cloud or on premises
- Deploy consistently across any cloud (Azure & AWS) or data center
- Run multiple Mule runtime versions in the same Runtime Fabric
- Isolate apps, scale horizontally, redeploy w/ zero downtime
- Connect to the control plane hosted by MuleSoft
- No need to dockerize Mule apps



Anypoint Runtime Fabric Components



All contents © MuleSoft Inc.

28

Comparing Anypoint Platform runtime plane options



Plane type	MuleSoft hosted in CloudHub	Runtime Fabric hosted	Customer-hosted
Managed by	<ul style="list-style-type: none">MuleSoft	<ul style="list-style-type: none">Customer	<ul style="list-style-type: none">Customer
Description	<ul style="list-style-type: none">iPaaS solution managed by MuleSoft	<ul style="list-style-type: none">Many CH-like runtime services provisioned into customer-hosted infrastructure	<ul style="list-style-type: none">Individual Mule runtimes are provisioned into customer-hosted infrastructureNo CH-like runtime services provided
Features	<ul style="list-style-type: none">Inbound HTTP load balancingVM queue messages are load balanced for deployments to multiple CH workers	<ul style="list-style-type: none">Inbound HTTP load balancingVM queue messages are load balanced for deployments to a cluster of Mule runtimes	<ul style="list-style-type: none">No inbound HTTP load balancing providedVM queue messages are load balanced for deployments to a cluster of Mule runtimes
Mule app SLAs	<ul style="list-style-type: none">Rescale Mule runtimes (CH workers) with zero downtime	<ul style="list-style-type: none">Rescale Mule runtimes with zero downtime	<ul style="list-style-type: none">None

All contents © MuleSoft Inc.

29

Comparing access to other Anypoint Platform runtime services based on the runtime plane type



Plane type	MuleSoft hosted in CloudHub	Customer-hosted (including Runtime Fabric hosted)
Object Store types	<ul style="list-style-type: none"> Connect to the Anypoint Object Store service with the Object Store connector In-memory OS 	<ul style="list-style-type: none"> Connect to the Anypoint Object Store service via REST API In-memory OS File-backed OS Hazelcast-backed OS
Anypoint MQ	<ul style="list-style-type: none"> Connect to AnypointMQ service with the AnypointMQ connector 	<ul style="list-style-type: none"> Connect to AnypointMQ service with the AnypointMQ connector (public internet access required)
Safely-hidden properties	<ul style="list-style-type: none"> Properties can be hidden from users of Runtime Manager 	<ul style="list-style-type: none"> Not available
Access security and restrictions	<ul style="list-style-type: none"> Mule apps can always access all CH services Security is automatically managed within MuleSoft managed infrastructure/networks 	<ul style="list-style-type: none"> Mule apps must have public internet access to these online services

All contents © MuleSoft Inc.

30

Deployment models for Anypoint Platform



Deployment type	CloudHub	Standalone Mule runtime	Anypoint Runtime Fabric	Anypoint Platform - Private Cloud Edition																					
Cloud-based	<table border="1"> <tr> <td>Design Center</td> <td>Exch.</td> <td>Mng Center</td> </tr> <tr> <td colspan="3">Mule runtime *</td> </tr> </table>	Design Center	Exch.	Mng Center	Mule runtime *			<table border="1"> <tr> <td>Design Center</td> <td>Exch.</td> <td>Mng Center</td> </tr> <tr> <td colspan="3">Mule runtime</td> </tr> </table>	Design Center	Exch.	Mng Center	Mule runtime			<table border="1"> <tr> <td>Design Center</td> <td>Exch.</td> <td>Mng Center</td> </tr> <tr> <td colspan="3">Mule runtime</td> </tr> </table>	Design Center	Exch.	Mng Center	Mule runtime						
Design Center	Exch.	Mng Center																							
Mule runtime *																									
Design Center	Exch.	Mng Center																							
Mule runtime																									
Design Center	Exch.	Mng Center																							
Mule runtime																									
Managed by	MuleSoft	MuleSoft	MuleSoft	MuleSoft																					
On-premises / private IaaS		<table border="1"> <tr> <td colspan="3">Mule runtime</td> </tr> </table>	Mule runtime			<table border="1"> <tr> <td colspan="3">K8s and Docker</td> </tr> <tr> <td>Design Center</td> <td>Exch.</td> <td>Mng Center</td> </tr> <tr> <td colspan="3">Mule runtime</td> </tr> </table>	K8s and Docker			Design Center	Exch.	Mng Center	Mule runtime			<table border="1"> <tr> <td colspan="3">K8s and Docker</td> </tr> <tr> <td>Design Center</td> <td>Exch.</td> <td>Mng Center</td> </tr> <tr> <td colspan="3">Mule runtime</td> </tr> </table>	K8s and Docker			Design Center	Exch.	Mng Center	Mule runtime		
Mule runtime																									
K8s and Docker																									
Design Center	Exch.	Mng Center																							
Mule runtime																									
K8s and Docker																									
Design Center	Exch.	Mng Center																							
Mule runtime																									
Managed by		Customer	Appliance: MuleSoft Infrastructure: Customer	Customer																					

* Includes load balancing, scaling, and zero downtime features

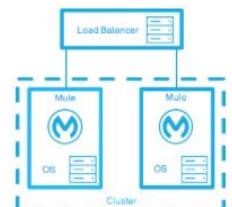
All contents © MuleSoft Inc.

31

- What control plane is used by Runtime Fabric?
- How do Runtime Fabric's runtime plane features and behaviors compare with the MuleSoft-hosted runtime plane
- How do Runtime Fabric's runtime plane features and behaviors compare with other customer-hosted runtime plane options?
- For which use cases is Runtime Fabric the best choice of runtime plane?

Clustered fault-tolerant deployment

- Only available for **customer-hosted runtime** planes
- A **cluster** is a set of Mule runtimes that acts as a unit
 - Servers in a cluster communicate and share information through a Hazelcast-based **distributed shared memory grid**, data is replicated across memory in different physical machines
- All worker nodes work in **active-active** model and always one (auto-elected)primary node
- Nodes are aware of each other
- If one node fails, outstanding tasks transfer automatically to surviving nodes
- Nodes can be added and subtracted from a cluster to adjust load capacity



Understanding Mule domains deployed to customer-hosted runtime planes



- A **Mule domain** project can be used to share global configuration elements between Mule applications, which can
 - Ensure **consistency** between Mule applications upon any changes, as the **configuration** is only set in one place
 - Expose multiple services within the Mule domain on the same port
 - Share the **connection** to persistent storage
 - Send events (messages) to other Mule applications using VM queues
- This allows Mule applications to **share resources** but NOT behaviours
- **ONLY** available for customer-hosted Mule runtimes, but not for Anypoint Runtime Fabric

How Mule domains are created



- Create a Mule Domain Project with a particular **domain name**
- Add global element configurations to the domain project
- Associate Mule applications with the domain name
 - Each Mule application automatically shares all the Mule domain's global elements

- How do customer-hosted Mule runtimes compare with an Anypoint Runtime Fabric runtime plane?
- Which scenarios are better suited for Anypoint Runtime Fabric, and which are better suited for customer-hosted Mule runtimes?
- Which control planes can control customer-hosted Mule runtimes vs. Anypoint Runtime Fabric?

Deciding deployment option for various scenarios



- **Regulatory** requirements of on-premises processing
 - Including metadata about API invocations and messages
- **Time-to-market**
- **IT operations effort**
- Accessing **on-premises data sources**
- **Flexibility of deployment across cloud providers**
- **Isolation** between Mule apps
- **Control over Mule runtime tuning**
- **Scalability** of runtime plane
 - horizontal and vertical; static and dynamic
- Roll-out of **new releases**
- **Redeployment with zero downtime**

- High availability (HA)
- Automated failover
- Out of box features required
 - Object store
 - Shared resource support
 - Persistent queues
 - Scheduling
 - Logging
 - Dashboards
 - Insights
 - Alerts
 - Auto scaling
- Anypoint Edge Security
- Tokenization

Deciding deployment option for various scenarios



- Technical notes for deciding deployment options are included in student files

Factors for deciding the best runtime plane to use based on organizational business goals



Runtime plane	CloudHub	Anypoint Runtime Fabric	Customer-hosted	Pivotal Cloud Foundry
Complex regulatory compliance required	Good	Maybe better	Maybe better	Maybe better
Time to market	Best	Better	Better	Maybe OK
Minimize IT operations effort	Best	Better	Good	OK

Deciding the best runtime plane based on performance and reliability goals



Runtime plane	CloudHub	Anypoint Runtime Fabric	Customer-hosted or Private Cloud Edition	Pivotal Cloud Foundry
Scalability	Auto scaling	Manual scaling	Manual scaling	
HA (multiple workers)	Yes	Yes	unless managed	Yes
Automated failover	Yes	Yes	unless managed	Yes
Redeployment with zero downtime	Supported	Supported	More work	Supported
Clustering of Mule runtimes	No	Yes*	Yes	Yes*

* With an external Hazelcast cluster service

42

Deciding the best runtime plane based on deployment flexibility goals



Runtime plane	CloudHub	Anypoint Runtime Fabric	Customer-hosted or Private Cloud Edition	Pivotal Cloud Foundry
Runtime tuning	Not supported	Not supported	Complete control	Template control
Upgrade to new Mule runtime version	Easy	Easy	More work	Moderate work
Access to on-prem data	VPC or over internet	Easier	Easier	Easier
Flexibility to deploy across providers	NA	Easy except GCP not supported	more work	NA

Deciding the best runtime plane based on state management goals



Runtime plane	CloudHub	Anypoint Runtime Fabric	Customer-hosted or Private Cloud Edition	Pivotal Cloud Foundry
Object Store service within the same Runtime plane	Supported	No	No	No
Persistent queues	Supported	Only in a cluster	Supported	Only in a cluster
Access to OSv2	Via an Object Store connector or the OSv2 REST API	Via the OSv2 REST API	Via the OSv2 REST API	Via the OSv2 REST API

Deciding the best runtime plane based on reusability goals



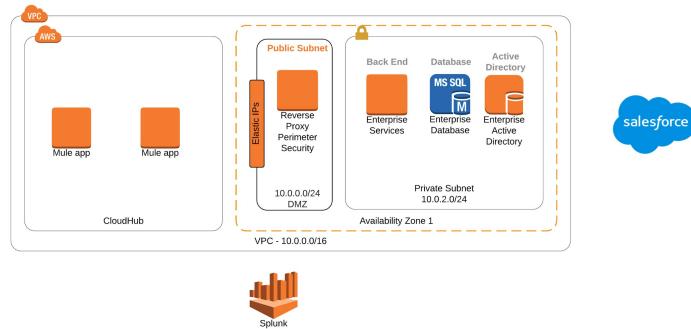
Runtime plane	CloudHub	Anypoint Runtime Fabric	Customer-hosted or Private Cloud Edition	Pivotal Cloud Foundry
Shared resources between apps	No	No	Yes (Mule domains)	No
Isolation between apps	Best (on VMs)	Better	From No to Best depending on configuration	Best (on VMs)

Exercise 7-1: Identify the best deployment model for a specific scenario



- Analyze deployment options for a specific scenario
- Design an optimal service model and deployment model for a specific scenario based on functional and non-functional requirements

ANYAIRLINE CURRENT PHYSICAL ARCHITECTURE



All contents © MuleSoft Inc.

46

Exercise scenario: Organizational requirements



- Data integration should have **minimal network latency**
- **Existing monitoring** capabilities of **Splunk** should be utilized
- **Existing HTTP-based services** should be utilized for cloud based **Salesforce** and **Splunk** systems
- The organization has **strict data residency requirements** and the **organization infrastructure** is **PCI compliant**
- Organization's **DevOps** has **limited expertise** in **containerization**

All contents © MuleSoft Inc.

47

- The organization has decided to use MuleSoft Anypoint Platform as the integration platform for data integration between various systems
- The Mule runtimes can use any available runtime plane that best meets the scenario's requirements
- High availability and scaling goals need to be supported
 - Mule applications should be able to be scaled "horizontally" across multiple Mule runtimes, so the services are highly available
 - Should be possible to scale the Mule runtime's host infrastructure "vertically" without service interruption

Exercise context: Organizational requirements

- The deployment model is influenced by below functional and non-functional requirements of the organization
 - HA and Scalability
 - Network latency
 - Monitoring
 - Networking/load balancing for endpoints
 - Security of data in rest/transit
 - Limited containerization capability of DevOps

Exercise step: Analyze deployment options for the scenario



Runtime plane	CloudHub	Anypoint Runtime Fabric	Customer-hosted or Private Cloud Edition	Pivotal Cloud Foundry
HA and Scalability				
Network latency				
Monitoring				
Networking				
Data security				
DevOps containerization capability				

Exercise step: Follow this decision tree to arrive at the best answer



- AWS autoscaling can be achieved using deployment on EC2 instances under ELB control
 - Can CloudHub, PCF, or Runtime Fabric be used to create these types of deployments?
 - If not, what runtime plane choices remain?
- What are the network latency requirements for this scenario?
- What is the fastest way to connect an on-prem network to a Mule runtimes in EC2 virtual images?

Exercise step: Deciding deployment options based on the type of control plane



Runtime plane	CloudHub	Anypoint Runtime Fabric	Customer-hosted or Private Cloud Edition	Pivotal Cloud Foundry
Scheduling				
CH enhanced logging				
Dashboard				
Insights				
Alerts				

Exercise step solution: Deciding deployment options based on the type of control plane



Runtime plane	CloudHub	Anypoint Runtime Fabric	Customer-hosted or Private Cloud Edition	Pivotal Cloud Foundry
Scheduling	Yes	No	Yes	Yes
CH enhanced logging	Yes	No	No	No
Dashboard	Yes	No	Yes	No
Insights	Yes	Basic support	Yes	No
Alerts	Yes	No	Yes	No

Exercise step: Decide deployment option for various possible scenario requirements



Runtime plane	CloudHub	Anypoint Runtime Fabric	On-prem runtime	Private Cloud Edition	Pivotal Cloud Foundry
Auto scaling					
Anypoint Monitoring					
Anypoint Visualizer					
Anypoint Edge Security					
Anypoint Tokenization					
Load balancing					
DLB support					
Anypoint MQ					
File Persistence					

All contents © MuleSoft Inc.

54

Exercise step solution: Decide deployment option for various possible scenario requirements



Runtime plane	CloudHub	Anypoint Runtime Fabric	On-prem runtime	Private Cloud Edition	Pivotal Cloud Foundry
Auto scaling	Yes	Manual scaling	unless managed	unless managed	unless managed
Anypoint Monitoring	Yes	No	No	No	No
Anypoint Visualizer	Yes	No	No	No	No
Anypoint Edge Security	No	Yes but minimal support	Yes	No	No
Anypoint Tokenization	No	No	Yes	No	No
Load balancing	Yes	Yes	No	No	No
DLB support	Yes	No	unless managed	unless managed	unless managed

All contents © MuleSoft Inc.

55

Exercise step solution: Deciding deployment option for various scenarios



Runtime plane	CloudHub	Anypoint Runtime Fabric	On-prem runtime	Private Cloud Edition	Pivotal Cloud Foundry
Anypoint MQ	Yes	No	No	No	No
File Persistence	Yes	Less reliable	Yes	Yes	Yes

All contents © MuleSoft Inc.

56

Exercise step solution: Analyze deployment options for the scenario



Runtime plane	CloudHub	Anypoint Runtime Fabric	Customer-hosted or Private Cloud Edition	Pivotal Cloud Foundry
HA and Scalability	Preferred	Preferred	Preferred	Preferred
Network latency	No (unless customer-hosted Mule runtimes are also on AWS EC2)	Preferred	Preferred	Preferred
Monitoring	No	Preferred	Preferred	Preferred
Networking	No	Preferred	Preferred	Preferred
Data security	No	Preferred	Preferred	Preferred
DevOps containerization capability	Preferred	Preferred	Preferred	No

All contents © MuleSoft Inc.

57

Exercise step: Deciding



- Using AWS auto-scaling requires deployment on EC2 instances under ELB control, which rules-out PCF, CH and RTF
- Hence only customer-manager "on-prem" Mule runtimes on EC2 remain
- Network latency demands fast IPSec or similar to on-prem network

Exercise step: Select runtime and control planes to meet requirements



- Which runtime and control planes can support AWS auto-scaling in EC2 instances under ELB control?
- What is the best way to minimize network latency between on-premises systems and the Mule runtimes running in EC2 under ELB control?

Exercise solution: Decision points



- Which runtime and control planes can support AWS auto-scaling in EC2 instances under ELB control?
 - PCF, CloudHub, and Runtime Fabric cannot support this requirement
 - The best choice is customer-managed Mule runtimes deployed into EC2 instances under ELB control
- What is the best way to minimize network latency between on-premises systems and the Mule runtimes running in EC2 under ELB control?
 - Fast IPSec or similar must be used to connect the EC2 instances with the on-premises network

Exercise step



- Decide the best two deployment options for the organization and summarize how each deployment model best serves organization requirement
- After some discussion, apply your analysis to identify the best deployment model for the scenario which meets the requirements

Exercise solution



- Customer-hosted Mule runtimes seem to meet all of the organization's requirements
 - Customer-hosted deployment minimizes network latency for data integrations
 - Customer-hosted runtimes in RTF or on-prem solution can use existing monitoring capability of Splunk
 - The data reside in customer environment so strict data residency requirement for organization is met
 - Customer-hosted runtimes in RTF or on-prem solution do not require DevOps with expert capability in K8s and Docker
 - No more outbound SSL endpoints as runtimes are deployed in existing infrastructure

Exercise solution



- Verify and discuss the differences between Anypoint Runtime Fabric vs. customer-hosted Mule runtimes, and decide if Anypoint Runtime Fabric is also a viable option

Summary



Summary



- Every Mulesoft service and deployment model provides different features
- Understanding organization ecosystem is key in deciding the deployment model
- Realization of key features for organization is important while deciding deployment model
- MuleSoft adds new features in products often and informs customer about its offerings



Module 8: Designing with Appropriate State Preservation and Management Options



Goal



Object Stores	Partitions	Keys	Values
Name	Partition	Key	Value
_defaultPers... 2592000	Token_store	globalKey	[binary value] BINARY
	_defaultPartition		
	_pollingSource_state-mo...		
	_pollingSource_state-mo...		
	aliastest		

Insight Logs	Name	Queued	In-Flight Transactions	Processed Messages
Object Store Queues	Test	1	0	Last 24 hours, updated every 5 min

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



- Decide the best way to store Mule application state in **persistent or non-persistent storage**
- Identify how to store Mule application state using the **Object Store v2**
- Decide the best way to manage storage of Mule application state using **persistent queues**
- Configure Mule application provided **caches** to store Mule application state
- Avoid duplicate processing of previous records using Mule connector **watermarks**

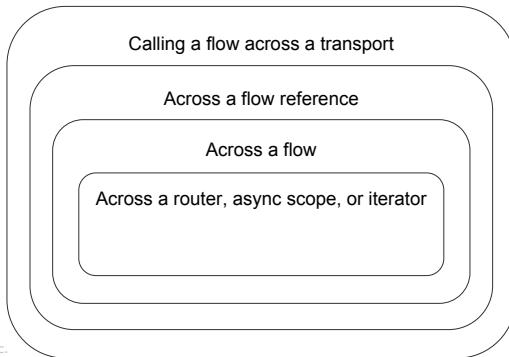
Distinguishing ways Mule applications can maintain state



The various ways the state of a Mule event may need to be available between event processors



- The Mule event stores state in the payload, attributes, and variables
- These can be passed between event processors and flows within various execution contexts



The various ways state may need to be shared beyond flow invocations

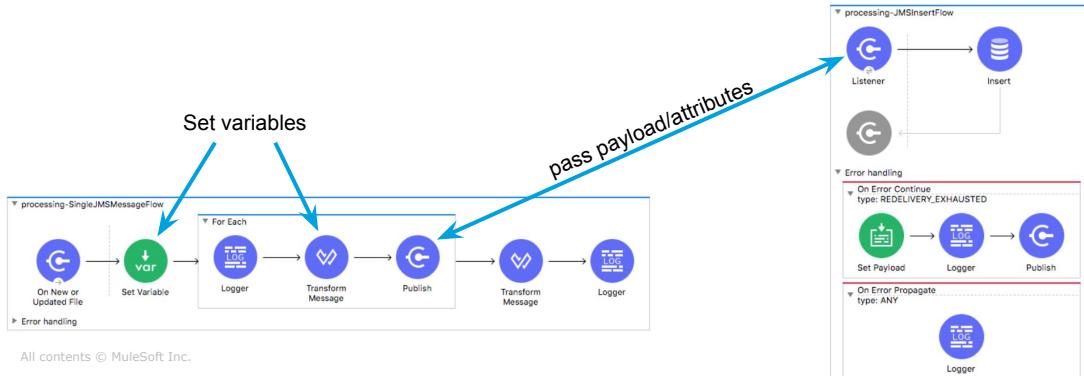


- Over time, state needs to be available to a particular event processor from other previous flow invocations or even other applications
- There are various levels of guarantees that can be coded and configured beyond one Mule event
 - When all the Mule runtimes restart
 - When the Mule runtime restarts
 - The next time a flow is invoked

Reviewing state management options



- You have already seen how to pass state
 - **Between** event processor **components** in a flow using variables
 - **Across transports** using attributes, payloads, or attachments
- This state is lost between subsequent executions of the flows



Use cases for saving state in a Mule application



- Between **flow executions**
 - To remember state from previous outbound HTTP requests in the flow
 - To filter out already processed messages for exactly once processing (idempotency)
 - To **cache** unchanged data or responses to speed up response time
- Between **iterations** (or **steps**) when processing a collection of elements
 - Including **batch processing**
 - Including iterations driven by a **scheduler** or **cron job**
 - For example, to aggregate results, such as a total price or total count
 - Each iteration (or batch step) may occur in parallel, perhaps in separate threads

Persistence guarantees that might be required by particular Mule applications



- Store state **non-persistently** in a single Mule runtime's memory
 - Data can be reused while the application is still running
 - For example, to retry an operation after a transient network glitch
 - Data is lost after application restarts or server crashes
- Store state **replicated** in a **distributed memory data grid cluster** between several Mule runtimes
 - Data survives application restarts or server crashes as long as one server with a replica survives
- Store state **persistently** to disk in a **single** Mule runtime
 - Data survives application restarts or server crashes, but not disappearance of the disk (such as when a CloudHub worker is stopped)
- Back a Mule runtime **cluster** with **persistent database storage**
 - Data survives if every server crashes

Persistence guarantees vs. performance



- Store state **non-persistently** in a single Mule runtime's memory
 - Fastest
- Store state **persistently** to disk in a **single** non-clustered Mule runtime
 - Added latency to write data to the local file
- Store state **replicated** in a **distributed memory data grid** or between several Mule runtimes
 - Added latency to replicate data across the network
- Back a Mule runtime **cluster** with **persistent database storage**
 - Added latency to replicate data across the network to the database
 - This may be the slowest option

When to maintain state in an external system or store

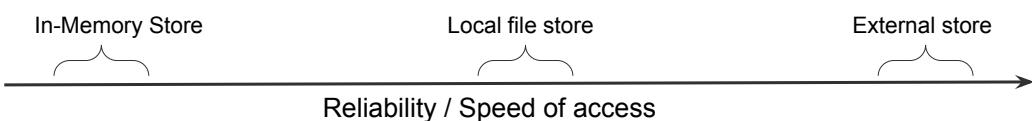


- An external system or store is required
 - When the Mule application requires additional guarantees, such as **transactional** guarantees
 - To share data among distributed applications, perhaps including non-Mule applications
- Examples include a **database**, an **object store**, or an **external cache**

Tradeoffs between different types of state storage



- Non-persistent in memory storage is the **fastest** way to access data, but is the least reliable/durable
- Local persistent storage, such as to a file system, is **slower** but more **reliable**, and not shared
- Replicated in-memory data grid storage lies somewhere between in-memory and on-disk
- An external storage system may be the most **reliable** and can be **shared**, but also may be much **slower**



Identifying MuleSoft object store behavior in various runtime planes



What is a MuleSoft Object Store?



- A MuleSoft **Object Store** is a key-value store implemented using Java
 - The values can be any serializable Java object
 - There is **no query mechanism**, objects are only retrievable by the key
- The interface definition of the MuleSoft Object Store is defined at
 - <https://github.com/mulesoft/mule-api/blob/master/src/main/java/org/mule/runtime/api/store/ObjectStore.java>
- From Mule application code, an Object Store is typically accessed via the Object Store connector
 - But an Object Store implementation might provide other communication mechanisms, such as a secure REST connection

- The Object Store component was designed to store state information between flow invocations
 - Synchronization information like watermarks
 - Temporal information like access tokens
 - User information
- Several factors change the runtime **behavior** or object stores
 - Whether an object store is configured as **persistent** or **non-persistent**
 - The runtime plane to which the Mule application is deployed
 - CloudHub worker(s)
 - A single customer-hosted Mule runtime
 - A cluster of customer-hosted Mule runtimes

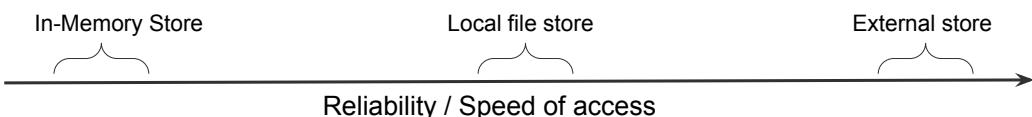
What MuleSoft means by persistence

- **Persistence** usually refers to storage that is copied to disk or some external storage, or replicated across several nodes
- **Non-persistent** usually refers to data that is only stored in the JVM memory of one Mule runtime
- Each object store can be configured as persistent or transient (non-persistent)

- **Durability** and **reliability** are more general ideas that refer to
 - The runtime plane to which the Mule application is deployed
 - One vs. multiple CloudHub workers
 - One vs. multiple vs. clustered customer-hosted Mule runtimes
 - Other tuning parameters like TTLs, storage limits, etc.
- All these interdependent factors affect the overall performance and SLAs of an objects store, so they must all be considered together to make good decisions

Behavior of various MuleSoft object store implementations based on storage type

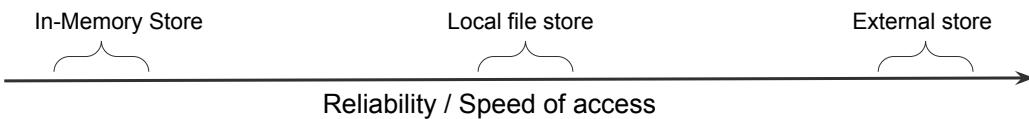
- Object store instances can be configured to be **persistent** or **non-persistent** in the Mule application
 - This parameters changes the runtime behavior, speed of access, and reliability of the object store



Behavior of various MuleSoft object store implementations based on storage type



- An object store's runtime implementation and behavior also varies based on the **runtime plane** and **version** and its **infrastructure**
 - Deployments vary between a single customer-hosted Mule runtime, single CloudHub worker, customer-hosted cluster, or multiple CloudHub workers
 - Customer-hosted clusters have additional configuration options that also affect the implementation and behavior



Choosing an object store instance to use in a Mule application



- Every Mule application is provided a **default** object store instance
 - This is automatically selected as the default object store for Mule components that use an object store
 - This object store is automatically configured as a **persistent** object store
- **Additional** object store **instances** (also called **partitions**) can be added as **global elements** to the Mule application

How CloudHub deployments implement a Mule application's object store



- A persistent object store uses the **Anypoint Object Store** service (OSv2)
 - A **cloud native implementation** of the Object Store interface
 - Data is shared between and accessible by all CloudHub workers
- A non-persistent object store does not use the OSv2 service
 - The object store is implemented locally in each CloudHub worker
 - Data is isolated within each CloudHub worker

Behavior of Mule 4 apps deployed via Anypoint Runtime Manager (RM) to various runtime planes



Runtime plane and its configuration	Object Store type defined in Mule app	Contents survives server restart (stop/start)?	Contents survives update to new app version (jar with different name)?	Contents shared between all nodes/workers?	Contents visible in RM?
Standalone runtime (previously configured via RM)	Transient Object Store	N	N	N/A	N/A
	Persistent Object Store	Y	N	N/A	N/A
Mule runtime cluster (previously configured via RM)	Transient Object Store	Y (1+ must remain)	Y	Y	N/A
	Persistent Object Store	Y (1+ must remain)	Y	Y	N/A
Multiple CloudHub workers, no OSv2, (persistent queues NOT ticked)	Transient Object Store	N	N	N	N
	Persistent Object Store	N	N	N	N
Multiple CloudHub workers, OSv2, (persistent queues NOT ticked)	Transient Object Store	N	N	N	N
	Persistent Object Store	Y	Y	Y	Y

Note: Application upgrade implies deploying a Mule application deployment package (jar) of a different name

When to code custom object store implementations



- In rare cases, custom object stores can be coded in Java
- This allows the Object Store connector to connect to a custom object store, such as an external database or data grid
 - <https://dzone.com/articles/custom-object-store-in-mule>
- An architect should evaluate the need to develop and maintain custom Java code, vs. reusing or coding a Connector to the external service, or using an external API directly

Configuring the storage behavior of object stores and VM queues in a cluster of Mule runtimes



- The object store is implemented in a Hazelcast data-grid
- The Hazelcast data-grid (the cluster) itself can be configured with different **store profiles**
 - With the default **reliable** store profile
 - VM queues are distributed across the data grid, providing HA and reliability behaviors
 - Object store are replicated across the data grid
 - With a **performance** store profile
 - Distributed queues are disabled, using local queues instead to prevent data serialization/deserialization and distribution in the shared data grid
 - The object store is implemented without replication to other nodes

Configuring the performance profile of a Mule runtime cluster



- A Mule runtime can set a performance or reliability profile
 - The default is reliable

```
mule.cluster.storeprofile=performance
```

- Each Mule application can override this setting in its configuration global element

```
<configuration>                                <configuration>
  <cluster:cluster-config>                    <cluster:cluster-config>
    <cluster:performance-store-profile/>      <cluster:reliable-store-profile/>
  </cluster:cluster-config>                  </cluster:cluster-config>
</configuration>                                </configuration>
```

Configuring the durability of a Mule runtime cluster



- A Mule runtime cluster can be manually configured in each Mule runtime's configuration
- A **quorum** size sets the minimum number of Mule runtimes that must be in the cluster in order Mule applications to run and accept inbound requests
 - This also sets the number of nodes to which data is replicated
- A JDBC store can also be configured to store cluster data from VM queues and object stores
 - So data survives restarting the cluster after all the nodes stop or crash

Reflection questions



- How is an object store implemented for a single Mule runtime?
 - What are the persistence options and how does this change the implementation?
- How is an object store implemented for a server group of Mule runtime?
 - What are the persistence options and how does this change the implementation?
- How is an object store implemented for a cluster of Mule runtime?
 - What are the persistence options and how does this change the implementation?
- How is an object store implemented for one or more CloudHub workers?
 - What are the persistence options and how does this change the implementation?
- What are the limitations to exchange data from each of these object stores?

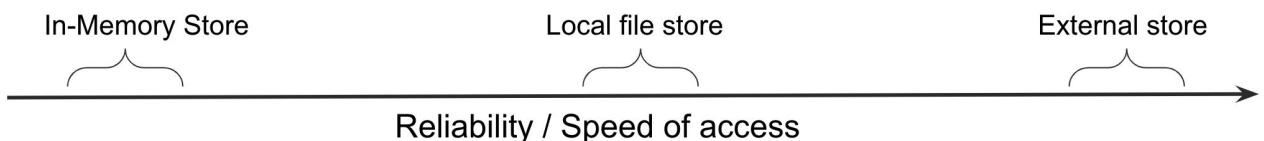
All contents © MuleSoft Inc.

30

Exercise 8-1: Identify why and when Mule applications need to maintain state



- Identify some scenarios where a Mule application needs to maintain state



All contents © MuleSoft Inc.

31

- Provide some example scenarios where state need to be maintained, and how is it maintained
 - Between applications
 - Between flows
 - Between HTTP requests
 - When schedulers are used
 - When a collection of records is processed
 - When a collection processes records out of order in batch processing
- What are the benefits and tradeoffs of storing state in these use cases
- How does the choice or runtime plane affect these decisions?

Exercise solution: When and why should Mule application state be maintained?

- To maintain the state of the Mule application between flow executions
 - To store a watermark (like the last file modified data) to avoid duplicate consumption of triggering data that has not changed
 - For example, to avoid duplicate synchronization between a source connector and target connector(s)
 - Multiple flow executions, triggered by different Mule events, perhaps asynchronously, but need to contribute to a common result
 - Such as aggregating (sum, average, ...) or accumulating (adding to list) values
 - Distinct incoming messages (such as HTTP requests) may contain duplicates, while avoiding duplicate processing

Exercise solution: When and why should Mule application state be maintained? (Cont.)



- To share variables between flows in a Mule application that are called by connectors rather than by flow references
 - However, it's more likely variables will be passed to the connector in the event payload or attributes
- To maintain authentication, session, or any other data which refreshes periodically or rarely
 - For example, to maintain a local lookup table of data or attributes that is populated from external sources that does not change often

Differentiating between the two Anypoint Object Store service versions



There are two versions of the Anypoint Object Store service



- Available in the MuleSoft-hosted runtime plane (CloudHub)
- Both versions store key/value pairs in an external online object store
- Both versions are secure, highly available online services
- Both versions can be used in a Mule application using the Object Store connector, when the Mule application is deployed to CloudHub
 - Mule 3 applications can also be configured to use OSv2
 - Mule 4 applications can only use the Anypoint Object Store v2 (OSv2)
- OSv2 also has a REST API for external applications to share object store data

Identifying the behavior of the Anypoint Object Store v2



- Simple key value store
- Synchronized access at key level
- Uses TLS for transport and FIPS 140-2 compliant encryption standard for data at rest
- Designed mainly for
 - Storing synchronization information
 - Storing short-lived information like access tokens
 - Storing user information
- Supports unlimited keys and 10 MB max value size
- Is preserved upon application redeployment and restarts
- Is deleted upon application deletion

- Not a universal data storage solution
- Does not support transactions
- Does not replace an actual DB and not suitable for searches, queries, etc.
- Data is automatically removed after a set amount of time
 - 30 days for OSv2

Creating and accessing MuleSoft object stores



- The **Object Store module** can be added to a Mule application or Mule domain
- This module is used to operate on object stores from the Mule application
 - Create new object stores as global elements
 - Use Object Store operations to store and retrieve data from an object store
- Depending where an object store is defined, it may be accessible with other Mule applications
 - An object store defined as a **Mule domain's** global element **can** be shared between Mule applications in that Mule domain
 - An object store defined as a **Mule application's** global element **cannot** be shared with other Mule applications

Deciding between object store types



Object store instances available to Mule applications



- Every Mule application is automatically provides a **default object store** that is **persistent**
- Other **global object stores** can be defined as global elements and used by any components in the Mule application or Mule domain
 - Used to isolate data access to a subset of components in the Mule application
 - Each global object store can have a different **partition name** to separate its stored data from other object stores used in the Mule application
 - Any additional persistent object stores use the same type of runtime storage as the default object store
 - Non-persistent object stores are always stored locally in the Mule runtime's memory

<https://blogs.mulesoft.com/dev/anypoint-platform-dev/new-mule-4-objectstore-connector/>

All contents © MuleSoft Inc.

43

How data is stored in the default object store



- You already saw that the persistence option is implemented differently depending on the runtime plane type
 - The runtime behavior also depends on the number of Mule runtime's to which the Mule application is deployed
- In the default object store, data is saved
 - To an online object store service when the Mule application is deployed to CloudHub
 - To shared distributed memory when the Mule app is deployed to a cluster of Mule runtimes
 - To a local file system when the Mule app is deployed to a standalone Mule runtime
- Mule applications deployed to customer-hosted infrastructure can use the OSv2 REST API to store data in OSv2 and retrieve it

All contents © MuleSoft Inc.

44

Blocking access to an object store from other Mule application components



- A **private object store** can be configured by a particular component, to securely hide its object store data from any other component in the Mule application
- Instead of configuring a global object store reference, certain components provide additional child elements to configure the object store
 - The object store configuration options (partition name, persistence, storage limits, etc.) are the same as for a global object store configuration

<https://blogs.mulesoft.com/dev/anypoint-platform-dev/new-mule-4-objectstore-connector/>

Ways to use various types of object stores to store state



- A global object store can share state
 - Between Mule components in a Mule application
 - Among the replicas of a Mule application executing on different nodes of a customer-hosted Mule runtime cluster
 - For OSv2, between the workers of a multi-worker CloudHub deployment of a Mule application
- Use a private object store when sharing data between components in the flow is deemed to be a security risk
- The Anypoint Platform Object Store v2 can also be used to share state between different Mule applications and distributed Mule runtimes using the OSv2 REST API

Reflection questions



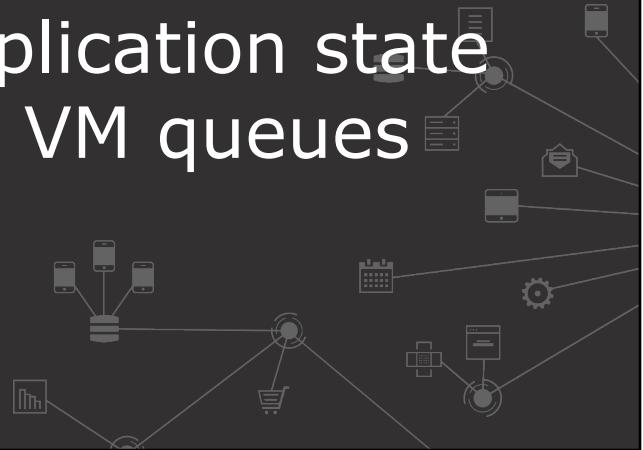
- What is the difference between a global object store and a private objects store?
- Which of these object store types can be persistent?
- Which of these object store types can be transient?
- What type of persistence is used by the default object store?
- How is the default object store implemented in each of the possible runtime planes?

Reflection questions



- What use cases are a good fit for using an object store?
- What use cases should not use an object store?

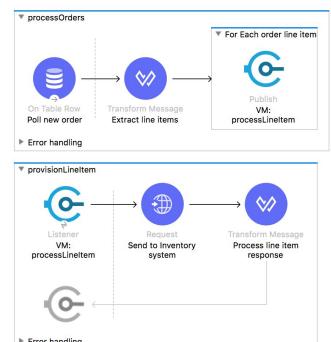
Storing Mule application state using persistent VM queues



How VM queues are used in Mule applications



- Mule applications can use VM queues
 - Each VM queue persists state in the Mule runtime's Java virtual machine in a first in first out (FIFO) order
 - Depending on the runtime plane, each VM queue may be shared
 - When a Mule application is deployed to a cluster of multiple Mule runtimes, the VM queue may be shared between all replicas of the Mule application
 - If the VM queue is in a Mule domain, several Mule applications on that Mule runtime can share the VM queue
- VM queues can be configured to persist after all Mule runtimes restart
- Other messaging systems can provide other SLAs and guarantees
 - Including sharing messages outside Mule apps



- Queues can be **transient** or **persistent**
- Transient queues are **faster** than persistent queues, but less reliable in case of a system crash
- Conversely, persistent queues are **slower** but more reliable
- For the customer-hosted runtime plane, queue persistence behavior also depends on the number of Mule runtimes and their dependencies
 - For standalone Mule runtimes, persistent queues are stored to the **local disk**
 - For a cluster of Mule runtimes, persistent queues are backed by the **cluster's distributed data grid**

VM queues behave differently based on the cluster's performance profile

- A cluster can be configured with a **reliable** or **performance** profile
- With a **performance** profile, VM queues and object stores do not use the data grid
 - Data is stored locally in the memory of the node in which the Mule application connects to the VM queue or object store
 - With multiple nodes, VM queue and object store data is split up and isolated

How persistent VM queues work in the MuleSoft-hosted runtime plane



- In CloudHub, persistent queues can retain certain Mule application data such as messages in VM queues after service outages
 - Slower, but more durable, than in-memory storage
 - Currently implemented using Amazon SQS storage
- The queue storage is automatically provided by the Anypoint Fabric service
 - Persistent queuing has no message limit
 - maxOutstandingMessages attribute is set to limit the number of messages saved in each VM queue

When and how to use persistent VM queues to share state



- Share messages (events) between the flows in a single Mule application
- Share messages (events) between Mule applications running in the same Mule domain(on-prem only)
- Share state (events) between Mule application instances deployed to multiple customer-hosted Mule runtimes (nodes) in a cluster or CloudHub workers

- All objects persisted must be serializable
- Overly complex structure may cause serialization error or performance issue
- Persistence queues may not guarantee that a message is delivered only once

Deciding to use persistent VM queues instead of another 3rd party messaging solution

- Use persistent queues when the use case does not require
 - Another more durable or more reliable state management option such as JMS or DB
 - Or to share state between multiple Mule applications or with non-Mule applications
- Persistent queues are not available
 - Between different Mule applications, especially deployed to different Mule runtimes
 - To non-Mule applications
- Persistent VM queues do **not** provide operation teams any advanced queue management features
 - Especially not in customer-hosted deployments

Managing state with file-based persistence



How file-based persistence works in customer-hosted runtime planes



- In **file-based persistence**, the data for state management is stored in files on the machine hosting the Mule runtime
 - Depending on the runtime plane and the Mule application's configuration, VM queues can also be persisted to files
- When a Mule application is deployed to **Customer-hosted** Mule runtimes
 - The account running Mule must have read and/or write permissions on the specified directories
 - VM queues can persist data in a file-based store
- File persistence is **less reliable, durable, and persistent** in Runtime Fabric compared to in a standalone Mule runtime

Do not use file-based persistence works for Mule applications deployed to CloudHub



- Mule applications deployed to CloudHub have **only limited** and **ephemeral** file system access
 - EC2 disk storage is removed when the CloudHub worker (EC2 instance) is removed
 - The File connector can only access specific folders such as /tmp or /opt/storage, depending on the CloudHub worker size

Limitation of file-based persistence



- File storage is typically less performant than in-memory storage
- File persistence does not work across nodes/workers/servers in replicated deployments of Mule applications
 - The file-system is not shared between CloudHub workers and also typically not within a Mule runtime cluster
- File storage is not transactional
 - Does not participate in transactions demarcated by a Mule application

Managing state with external storage systems



How external state management works



- Mule applications can also use connectors to external state management systems
 - Data is stored in a database, cache, or a cache backed by a database or API
- Different external stores provide various **quality of service** levels
 - Persistence
 - Transactional
 - Replication
 - Various eviction policies (Least frequently used)
 - High availability through cluster
 - Fast data retrieval through partitioning
 - Automatic failover
- Various options support a wide range of data structures
 - String, Maps, List, Set, Blob dependent on data stores
- MuleSoft has connectors for Redis and MongoDB

When to use an external store to store and manage Mule application state



- Pros
 - May provide more exacting performance, reliability, and durability guarantees
 - May be a preferred option to achieve certain high availability or failover goals
 - Use when backup and replication is required for cache objects
- Cons
 - Additional cost, management, and staffing requirements
 - Added layers and complexity

Limitation of external store



- Need to maintain one more system for state management
- High network latency as a result of network call to external system

Behavior of Mule 4 apps deployed via Anypoint Runtime Manager (RM) to various runtime planes



Runtime plane and its configuration	Queue type defined in Mule app	Messages survive server restart (stop/start)?	Messages survive update to new app version (jar with different name)?	Messages shared between all nodes/workers?	Messages visible in RM?
Standalone runtime (previously configured via RM)	Transient queue	N	N	N/A	N/A
	Persistent queue	Y	N	N/A	N/A
Mule runtime cluster (previously configured via RM)	Transient queue	Y (1+ must remain)	Y	Y	N/A
	Persistent queue	Y (1+ must remain)	Y	Y	N/A
Multiple CloudHub workers, no persistent queues, (OSv2)	Transient queue	N	N	N	N
	Persistent queue	N	N	N	N
Multiple CloudHub workers, persistent queues, no queue encryption, (OSv2)	Transient queue	Y	Y	Y	Y
	Persistent queue	Y	Y	Y	Y

Note: Application upgrade implies deploying a Mule application deployment package (jar) of a different name

All contents © MuleSoft Inc.

65

Designing Mule applications that use the Cache scope



- The Cache scope maintains the state of **multiple requests** in a Mule application over a particular time frame
- Avoid **duplicate** processing
- Maintains the state of requests in and **in-memory or persistent object store, or in an external third party store**

Limitation of caching

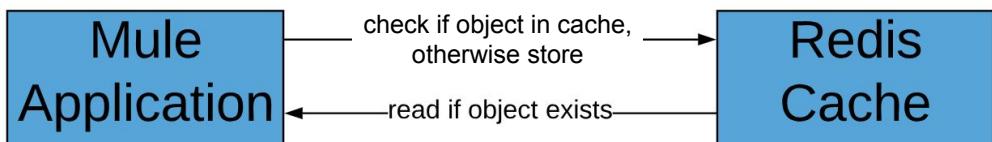
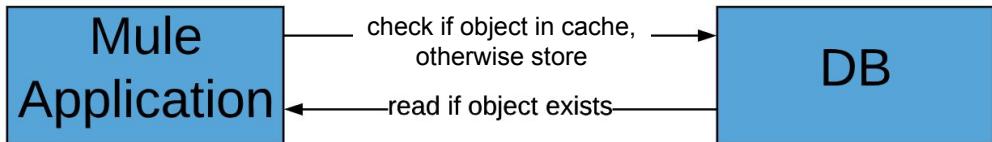
- Does not cache consumable payloads such as a stream
- An in-memory cache is fastest but least persistent
- In Mule 4, the cache scope does not directly support external stores such as DB, Redis
 - Can be done using the Mule SDK

- What use cases should use a local cache?
- What use cases should use a cache backed by an external store?
- What types of external stores are possible, and how do you decide the best choice?
- What use case would benefit from an intermediary API between caching operations in the Mule application and an external store?

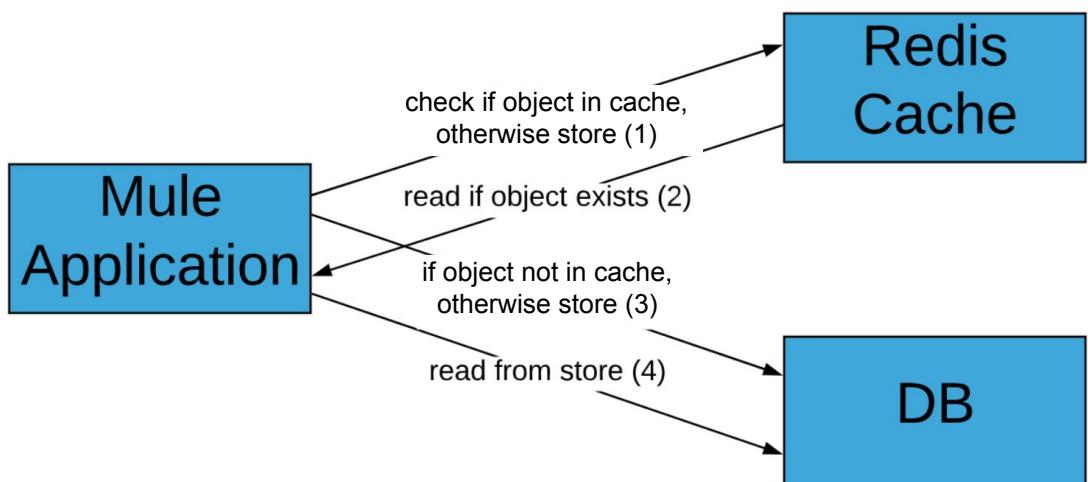
Designing with an external cache

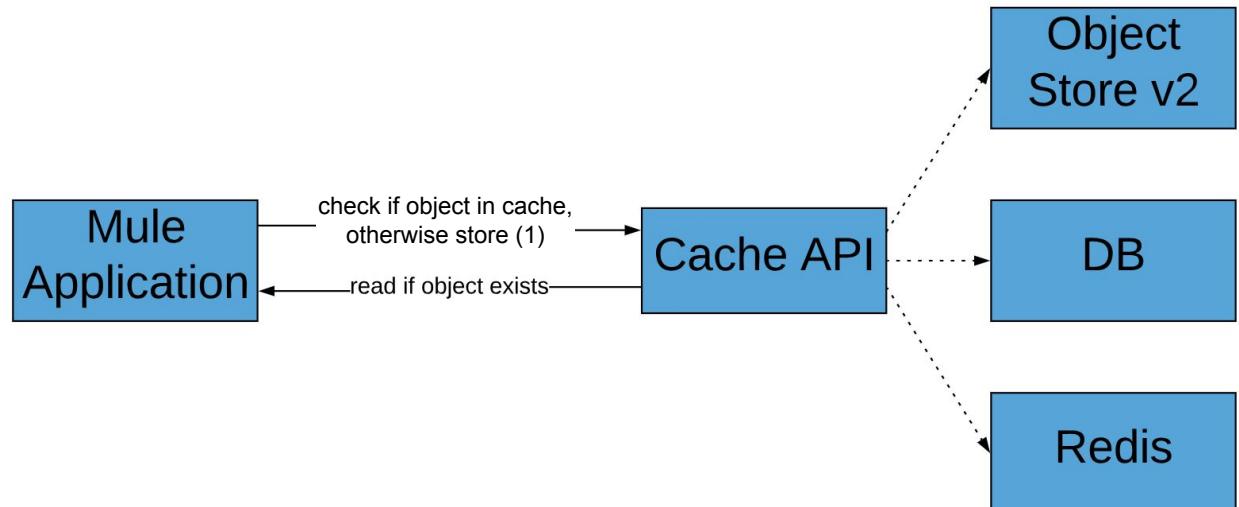


Configuring Mule applications to use a local cache plus external storage



State management using cache backed by store





Avoiding duplicate processing using watermarks



What is a watermark?



- A **watermark** is a form of state (in other words, a value) that is stored during a recurring processing cycle, such as to process a collection of records
 - During the next processing cycle, the watermark is retrieved and compared against the new corresponding values in the next processing cycle
 - Only records with a new value **bigger** or **higher** than the previous watermark are processed
- Examples
 - The timestamp of a previously processed files
 - The last account id processed in a group of records, where the account id is **always increasing**

Types of watermarks typically used in applications



- **Automatic**
 - The saving, retrieving, and comparing is automatically handled for you through an object store
 - 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

- There is a watermarking option for the **On New and Updated File** operation for the family of file connectors
 - There are two watermarking modes
 - CREATION_TIMESTAMP
 - MODIFIED_TIMESTAMP
- The Database connector has watermarking option an **On Table Row** operation that is triggered for every row in a table
 - On each poll, the component will go through all the retrieved rows and store the maximum value obtained

Limitation of watermarking

- Not all connectors supports watermarking
- Does not supports **multiple columns** in DB connectors
- Watermark attributes are limited for each connector
- Asynchronous processing may not deliver the watermarked value in the correct increasing order, which might cause new records to be skipped

- How does an On Table Row operation's watermark work?
 - How does this feature compare with using a Scheduler and any other database operation?
 - How does an On New or Updated File operation's watermark work?
 - How does this feature compare with using a Scheduler and any other database operation?

Deciding the best state storage and state management options



Deciding state management options for a specific use case



- Some factors involved when deciding the best state storage and management options for a use case
 - What are the performance goals and how can state storage options help meet those goals?
 - Can caching avoid duplicate processing?
 - What part of requests or response events should be stored or cached?
 - Does stored data need to be encrypted, and if so what are the tradeoffs?

Exercise 8-2: Design state management for a polling use case



- Identify ways to improve performance by storing and managing the state of Mule events in a flow
- Identify ways to avoid duplicate data processing by storing and managing the state of Mule events in a flow
- Decide when the state of response data should be stored
- Decide when stored state data should be encrypted
- Identify the best integration style for a scenario

- A scenario that may require state management
 - The mule flow is **polling a backend system API multiple times**
 - The **backend** system API call is takes **over 30 seconds** to respond
 - The **response** from the **backend** system contains **PII (Personally Identifiable Information) data**
 - Sensitive data should be encrypted and secured within the Mule application
 - What is best solution to **improve performance** by managing state from the flow
 - Architect your integration solution to best fix this particular use case

Exercise step: Compare state management options in the context of the deployment model

- Compare state management options based on performance and reliability goals in context of the deployment model

Runtime Plane	Object store v1	Object store v2	Persistent queue	File store	Cache	Watermark
Access Control						
Persistence lifespan						
Cluster Support						
Has REST API						

Exercise step: Decide the best state management option based on performance and reliability goals



Runtime Plane	Object Store v1	Object Store v2	Persistent Queue	File Store	Catch	Watermark
Automated failover						
HA (multiple workers)						
Transactional						
Encryption						
Performance						
Replication						

All contents © MuleSoft Inc.

87

Exercise steps



- Decide the following questions
 - How can you **improve the delayed response time** cause by the API taking more than 30 seconds to respond?
 - Is there a way to **avoid calling the same API repeatedly**, and what are the benefits of avoiding these repeated API calls?
 - Is **state management needed**, and if so, why?
 - How can state management help this use case?
 - If state management is needed, **what** Mule application **state or objects should be stored** (Mule event, request and response)
 - If you design to avoid repetitive API calls, then how should the Mule application **send back its response**?

All contents © MuleSoft Inc.

88

Exercise steps



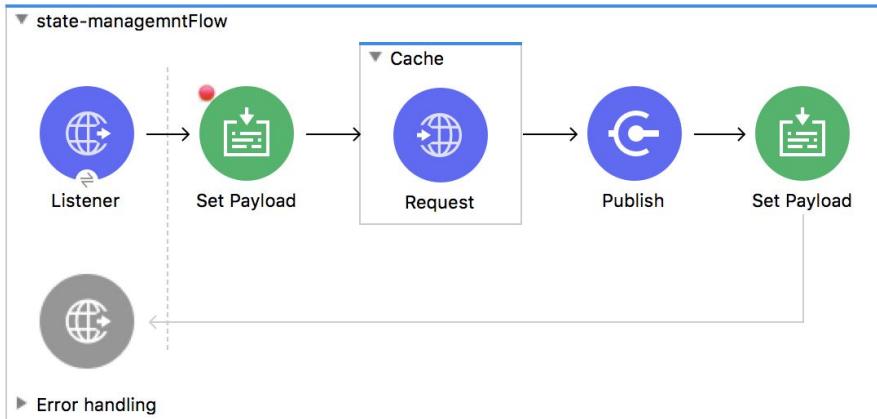
- Decide these follow up questions
 - Should responses that were sent back to clients be stored?
 - When should state management objects be stored?
 - What kind of state management option should be used?
 - File, Persistent Queues, Object Store v2, Caching, Watermark
 - Can PII data be stored in the Mule application's store?
 - Must data be encrypted in the store, and if so, how?
 - Should the component using the store restrict access from other components in the Mule application?
- Agree with the rest of the class on the requirements for this use case

Exercise steps



- Use Anypoint Studio to mock a flow to meet the agreed requirements
- After everyone completes a solution, discuss and compare all the solutions

Exercise solution



Error handling

```
<ee:object-store-caching-strategy name="Caching_Strategy" >  
    <os:private-object-store alias="aliastest" />  
</ee:object-store-caching-strategy>
```

Summary



- Applications often require various persistence guarantees to store state
- Mule applications can leverage various features to more easily meet persistence guarantees
- **Object Stores** persist and share a watermark (or other data) across flow executions
- Use persistent queues for managing state of application in case of failure of application or Mule runtime
- Use Caching to avoid intensive processing for repetitive payload
- Persists cache objects in object store to share across requestsUse a watermark to keep a persistent variable between scheduling events

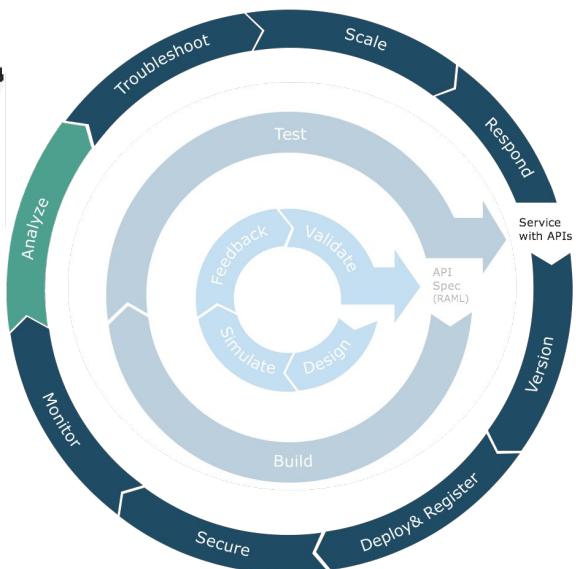
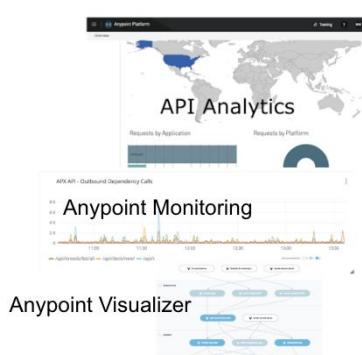
- Applications often require various persistence guarantees to store state
- Mule applications can leverage various features to more easily meet persistence guarantees
- Use the **Object Store** connector to persist and share a watermark (or other data) across flow executions
- Use persistent queues for managing state of application in case of failure of application or Mule runtime
- Use Caching to avoid intensive processing for repetitive payload
- Persists cache objects in object store to share across requestsUse a watermark to keep a persistent variable between scheduling events



Module 9: Designing Effective Logging and Monitoring



Goal



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



- Identifying auditing options for Mule application
- Identify the logging strategy for a Mule application
- Decide Mule application logging options
- Analyse integration options with third party log management system
- Decide monitoring, alerting, and notification options
- Determine API reporting options

Identifying auditing options for Mule application



- **Access Management** contains the **audit log**
 - Logs user interactions within Anypoint Platform, including logins, creating business groups, and creating environments
 - Does NOT include system or Mule application tracking and tracing
 - The organization owner can access all audit logs
- Audit logs for a particular business group can be viewed for all users assigned the Audit Log Viewer role for that business group
- The audit log is primarily used to detect **access violations**

Persisting audit logs

- Keeps a permanent queryable history of user activities in an Anypoint Platform organization
 - Retained for six years
 - Periodically download these audit log files to keep them longer
- View and download the audit logs from the audit logging service
 - Use the Audit Logging Query API
 - View through the Anypoint Platform Access Management UI

Identifying logging options for Mule application



Logging in Mule



- Logs help to **debug and track** processing of Mule applications, such as details from Mule events
- In Mule flows, the Logging component creates log messages with various log levels, such as INFO, WARN, ERROR, and DEBUG
- Log messages can be configured to log to various **appenders**
 - The system console, a file, a database, the CloudHub logging service, or another server or service
- By default, Mule runtimes show the INFO log level, so ignore DEBUG or TRACE level log messages

Mule runtimes and Mule applications use standard slf4j and log4j2 logging



- Supports synchronous or asynchronous logging
- By default, logging in Mule is done **asynchronously**
- Can be configured with standard log4j2.xml at the system and Mule application level

System vs Application logs in the CloudHub runtime planes



System log

- Specific to the Mule runtime
- Configured by log4j2.xml inaccessible to customers
- Contains
 - Log messages about the Mule runtime lifecycle (startup and shutdown)
 - Status messages about Mule application

Application log

- Specific to the Mule application
- Configured by log4j2.xml typically packaged in Mule application
- Contains
 - All log messages generated inside the Mule application
 - Including System.out messages

System vs Application logs in customer-hosted runtime planes



System log

- Specific to the Mule runtime
- Configured by log4j2.xml inaccessible to customers
- Contains
 - Log messages about the Mule runtime lifecycle (startup and shutdown)
 - Status messages about Mule application **and Mule domain deployments**
 - **System.out messages**

Application log

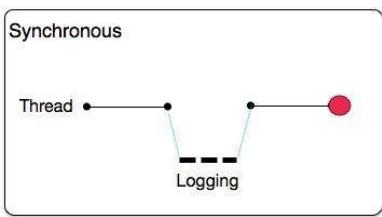
- Specific to the Mule application
- Configured by log4j2.xml typically packaged in Mule application
- Contains
 - All log messages generated inside the Mule application
 - Including `System.out` messages

Synchronous vs asynchronous logging



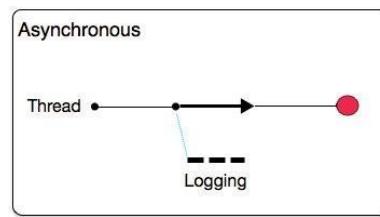
Synchronous

- The execution of the thread that is processing your message is interrupted to wait for the log message to be fully output before it can continue
- Performance degrades because of synchronous logging
- Used when the log is used as an audit trail or when logging ERROR/CRITICAL messages



Asynchronous

- The logging operation occurs in a separate thread, so the actual processing of your message won't be delayed to wait for the logging to complete
- Substantial improvement in throughput and latency of message processing
- Log may be lost in case of system crash



Configuring custom logging settings for a Mule runtime and its Mule applications



- The default log setting for Mule is asynchronous and at a level greater than or equal to INFO
 - So DEBUG and TRACE level messages are ignored
- Runtime Manager can override the log level for a deployed Mule application without redeployment

Runtime	Properties	Insight	Logging	Static IPs
Additional log levels and categories to include in logs. Learn more about logs.				
DEBUG	▼	com.mulesoft	X	
INFO	▼	org.mule	X	
DEBUG		package.name		
WARN				
ERROR				
INFO				

[Apply Changes](#)

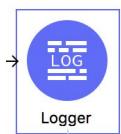
All contents © MuleSoft Inc.

13

The Mule Logger component



- If nothing is specified in the message attribute of the Logger component, the entire **Mule event** is logged, including all attributes and variables
 - A Logger does **not** log the contents of the **payload**, only its type
- A Logger sets the content of the message and the log level



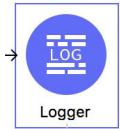
```
00:23:46.903      11/15/2018     Worker-1      [MuleRuntime].cpuLight.01: [mule-objectstore-prod-am].readHtmlFiles.CPU_LITE
@699e1bae      DEBUG
ChildEventContext { id: 0-c33d5ea0-e8af-11e8-b463-0262da9ac180_296955003_2025473580; correlationId: 0-c33d5ea0-e8af-11e8-b463-
0262da9ac180; flowName: help; componentLocation: help/processors/0; response completed with result.
```

16

Using log messages to trace messages



- MuleSoft recommends logging messages with a unique id to help to trace the Mule event flow (the transaction) through the system
 - A unique ID could be
 - The correlation id that is automatically set in the Mule event
 - A system generated ID stored in the payload or HTTP request header
 - Another ID value stored in the payload or HTTP request header
- The correlation id is automatically logged
 - For both CloudHub and customer-hosted Mule runtimes, the Logger component automatically logs the correlationId for DEBUG level messages



```
00:23:46.903      11/15/2018     Worker-1      [MuleRuntime].cpuLight.01: [mule-objectstore-prod-am].readHtmlFiles.CPU_LITE
@699e1bae      DEBUG
ChildEventContext { id: 0-c33d5ea0-e8af-11e8-b463-0262da9ac180_296955003_2025473580; correlationId: 0-c33d5ea0-e8af-11e8-b463-
0262da9ac180; flowName: help; componentLocation: help/processors/0; response completed with result.
```

17

Log retention in CloudHub



- CloudHub truncates log messages
 - Up to 100 MB per Mule application and per worker
 - At most 30 days,

The screenshot shows the 'Deployments' section of the CloudHub interface. It displays a deployment from 'Today' at 12:36. Under the 'System Log' tab, there are two workers listed: 'Worker-0' and 'Worker-1'. Each worker has a 'Logs' button with a download icon. The 'Logs' button for 'Worker-0' is highlighted with a blue border, indicating it is selected. Below the log buttons, there is a 'Diagnostics' button with a download icon. At the bottom of the log section, there is a message: '> 12:32 - Deployment'.

Choosing between custom logging options



How CloudHub configures logging for a deployed Mule application



- By default, CloudHub replaces a Mule application's log4j2.xml file with a CloudHub log4j2.xml file
- The CloudHub log4j2.xml file
 - Specifies the CloudHub appender to write logs to the CloudHub logging service
 - Sets default log levels for various log categories
 - These levels can be overridden in Runtime Manager in the Mule application's Logging configuration
- There is a process to ignore the default CloudHub log4j2.xml file

Sending CloudHub logs to external logging systems using a **custom log appender**



- In CloudHub, you can disable the CloudHub provided Mule application log4j2 file
 - Allows integrating Mule application logs with custom or third-party log management systems
 - Only available on request via the support portal
 - Once enabled, the Mule application's log4j2.xml file is used
 - Can send/export application logs to other log4j2 appenders, such as a custom logging system
 - The system logs cannot be exported to other appenders
 - MuleSoft is not responsible for lost logging data due to misconfiguration of your own log4j appender
 - Mule application logs may not appear in Runtime Manager anymore and may no longer be available for download

Combining a CloudHub log appender with custom log appenders



- The CloudHub appender can also be configured inside a Mule application
 - Then the CloudHub appender will also be used if the default CloudHub log4j2.xml file is disabled for the Mule application deployment
 - Then Mule application logs will still appear in Runtime Manager and can be downloaded
 - See the docs for configuring both the **CloudHub log appender and custom log appenders at the same time**
 - Reference: Custom-CloudHub-log4j2-file.xml in the Module 9 student files

CloudHub can integrate with third party log management systems with a **custom aggregator**



- As another option, you can create a **custom aggregator application**
 - A custom application to fetch logs using CloudHub APIs and then send the logs to an external system
 - Can be done by creating an application with Mule or any other language
 - The custom aggregator application must programmatically recognize or discover new Mule runtimes (nodes) used by Mule application deployments
 - For example, when a new customer-hosted Mule runtime is added to a server group or cluster

How to integrate with third-party log management systems from customer-hosted runtime planes



- Use the Splunk or ELK plugins in Runtime Manager
 - Uses the Runtime Manager agent to exchange logs and analytics data with Splunk or ELK
 - Plugins to other third-party tool can be developed
- Fetch the logs from the third-party server side
 - Configured from the specific logging system server side to take/read the log files directly from the Mule server
 - For example, using the Splunk Universal Forwarder
- Add log4j-specific system appenders to the Mule runtime
 - Specific system appender libraries are placed in \${MULE_HOME}/lib/user

Comparing logging retention and storage limits for various runtime planes



- CloudHub logging limits the log retention period and log size
- Customer-hosted runtime can provide large storage space for logs
 - But is still limited by the available storage of the host or VM

Comparing the ability to externalize logs for various runtime planes

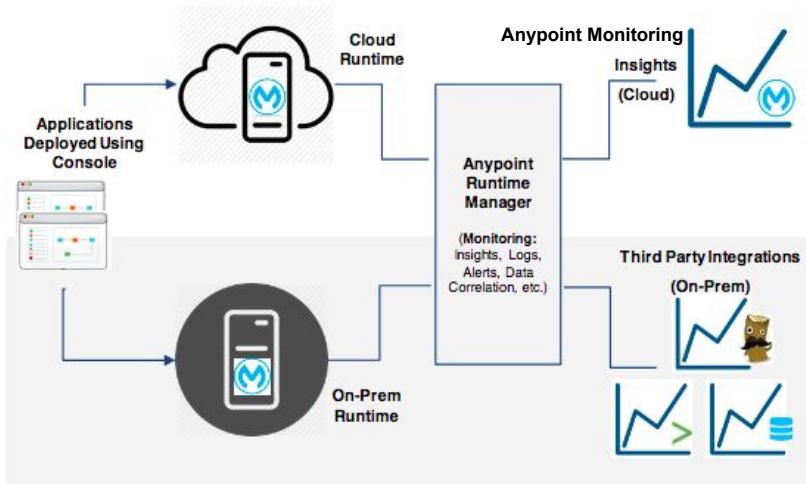


- In Mule applications deployed to CloudHub, the Mule application log messages can be sent to external log management systems
 - But the Mule application **log may be lost if the external log system fails**
 - Or if network connectivity is lost between CloudHub and the external log system
- CloudHub system log messages **cannot** be sent to external log management system without installing custom CH logging configuration through support
- Customer-hosted runtime can **send system and application log** to external log management system

- What scenarios would require some or all Mule application logs to be externalized?
 - What scenarios would not allow storing Mule application logs in CloudHub, and what are the tradeoffs resulting from this decision?
 - What is the tradeoff when Mule application logs are only stored in CloudHub?

Choosing monitoring, alerting, and notification options





Monitoring applications

- Anypoint Platform involves different types of capabilities when dealing with monitoring
 - Anypoint Monitoring
 - Anypoint Analytics
 - Runtime Manager Dashboard stats
 - CPU, memory, mule messages
 - Business events Insight
 - Default
 - Custom

Monitoring Mule applications with the Anypoint Monitoring dashboard



Monitoring applications using the Anypoint Monitoring dashboard



- Gives insight into application usage and performance
 - <https://docs.mulesoft.com/monitoring/dashboards>
- Helps to view dashboards (both overview and custom)
- Can be accessed by Organization Administrators and API Version Owners/API Creators

The screenshot shows the Anypoint Monitoring dashboard interface. At the top, there are dropdown menus for 'Environment' (set to 'Sandbox') and 'Resource' (set to 'object-module.us-e2.cloudhub.io'), and a time range selector set to '30 minutes'. Below this is a navigation bar with tabs: 'Overview' (which is active and highlighted in blue), 'Inbound', 'Outbound', 'Performance', 'Failures', 'JVM', and 'Infrastructure'. The main content area displays an 'Overview' card for the application 'object-module.us-e2.cloudhub.io'. The card includes the following details:

Type:	CloudHub Application	Runtime:	4.1.2-AM
Status:	Started	Worker:	0.1 vCores x 2
Region:	us-east-2	Last updated:	33 minutes ago
Domain:	object-module.us-e2.cloudhub.io	Manage Application	

A 'Hide Detail' button is located in the top right corner of the card. At the bottom of the dashboard, there is a footer note: 'All contents © MuleSoft Inc.' and a page number '33'.

Monitoring applications using the Anypoint Monitoring **built-in dashboard**



- Default build-in dashboard in Anypoint Monitoring contains a set of time series charts to collect various metrics
 - Inbound and Outbound events
 - Other performance metrics
 - API functional monitoring
- Complete details of each dashboard are mentioned at <https://docs.mulesoft.com/monitoring/dashboards>

Built-in dashboard **traffic** metrics

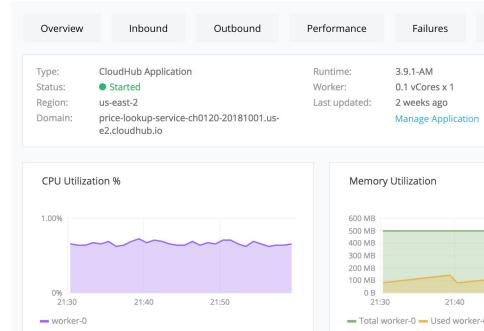


- Inbound
 - Total inbound requests, Avg. response time inbound, Total inbound requests by endpoint, Avg. response time inbound by endpoint, Total inbound requests failed
- Outbound
 - Total outbound requests, Avg. response time outbound, Total outbound requests by endpoint, Avg. response time outbound by endpoint, Total outbound requests failed
- Performance
 - Avg. response time inbound, Avg. response time outbound, Avg. response time inbound by endpoint, Avg. response time outbound by endpoint
- Failures
 - Total inbound requests failed, Total outbound requests failed, Total inbound requests failed by endpoint, Total outbound requests failed by endpoint

Built-in dashboard **infrastructure** metrics



- JVM
 - GC count, Memory stats (heap and meta), Thread count
- Infrastructure
 - CPU, Memory utilization, Thread count, Total system memory and total system processor



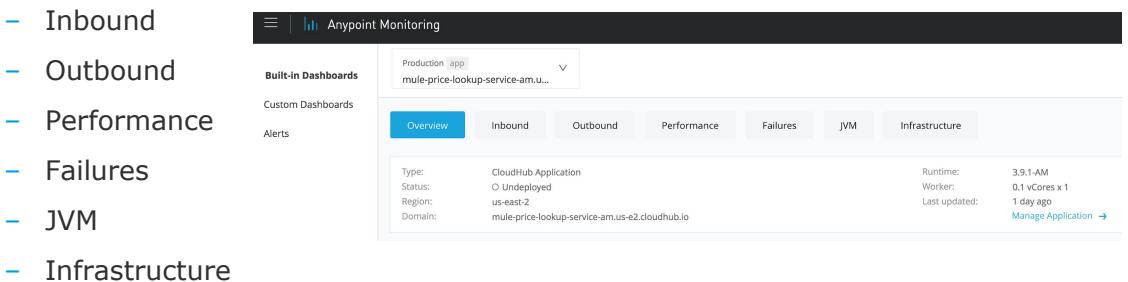
All contents © MuleSoft Inc.

36

Exercise 9-1: Explore the Anypoint Monitoring dashboard



- Login to anypoint.mulesoft.com and go to Anypoint Monitoring
- Select environment and resource for viewing monitoring details
- Review an resource usage and performance using below Anypoint Monitoring dashboards



All contents © MuleSoft Inc.

37

Reflection questions



- What type of information is not available in the Anypoint Monitoring dashboard?
- What are the use cases and requirements for this missing information?

Monitoring applications using Anypoint Functional Monitoring



- Enables developers and operators to perform consistent testing of the functional behavior and performance of an API

The screenshot shows the Anypoint Monitoring interface. On the left, there's a sidebar with options like 'Built-in Dashboards', 'Custom Dashboards', 'Alerts', and 'Functional Monitoring'. The 'Functional Monitoring' option is selected. The main area is titled 'Health Check' and displays a chart for 'Version 1.0.0'. The chart has a y-axis from 0 to 18 and an x-axis labeled 'Mon'. It shows three bars: one red bar at approximately 12, one purple bar at approximately 10, and one green bar at 1. Below the chart, there's a table with three rows:

Test	Last run	Endpoints status
test1	34 seconds ago	1/1
Health Check	2 minutes ago	1/1
test	2 minutes ago	1/1

Each row includes a green progress bar indicating the number of endpoints tested. To the right of the chart, there's a summary section with the status 'PASSED' and details: 'Last run 16 seconds ago', 'Test duration 4s', 'Execution Location us-east-1', and 'Environment default'. There are also icons for trash, edit, and refresh.

- Write tests manually and then schedule them with the Black box Automated Testing (BAT) CLI
 - Test cases are based on the DataWeave language and follows the “given-when-then” approach to describe behavior in terms of conditions and expected outcomes
- Create monitors in the **Functional Monitoring** section of Anypoint Monitoring

Monitoring APIs using the Anypoint Analytics dashboard



Monitoring APIs using the Anypoint Analytics Dashboard



- Gives insight into API usage and performance
 - <http://anypoint.mulesoft.com/analytics>
- Helps to view dashboards (both overview and custom), create and manage charts and reports
- Can be accessed by Organization Administrators and API Version Owners/API Creators

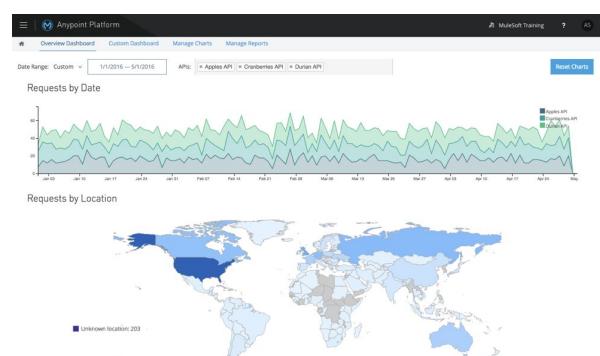
The screenshot shows the Anypoint Analytics interface. On the left, there's a sidebar with 'API Administration (Sandbox)' and tabs for 'SANDBOX', 'API Administration', 'Client Applications', 'Custom Policies', and 'Analytics'. Below that is a 'All content' section. The main area has a 'Manage API' button, a search bar, and a table with columns: API Name, Version, Status, Client Applications, and Creation Date. One row is visible for 'American Flights API' with version 'v1'. A detailed view of 'v1' shows two instances: 'v1:11435814' (Active) and 'v1:15522939' (Unregistered). To the right, a panel for 'American Flights API v1' displays 'Manage CloudHub Proxy', 'View API in Exchange', and a large 'View Analytics Dashboard' button with an arrow pointing to it. Below this are tabs for 'Applications', 'Policies', and 'SLA tiers', and a note: 'There are no applications for this API version.' The page number '42' is in the bottom right.

Viewing API analytics



- The default dashboard contains a set of charts

- Requests by date
 - Line chart representing number of requests
- Requests by location
 - Map chart showing the number of requests for each country of origin
- Requests by application
 - Bar chart showing the number of requests from each of the top five registered applications
- Requests by platform
 - Ring chart showing the number of requests broken down by platform



Exercise 9-2: Explore the Anypoint Analytics Dashboard



- Login to anypoint.mulesoft.com and go to API Manager
- Navigate to Anypoint Analytics
- Review an API's usage and performance using the API Analytics Dashboard
- Explore the Overview Dashboard

Custom API dashboards



- The **custom dashboard** in Anypoint Analytics contains a set of charts for a single API or for all APIs
- Each chart displays various API characteristics
 - Requests size
 - Line chart representing size of requests in KBs
 - Requests
 - Line chart representing number of requests over a period
 - Response size
 - Line chart representing size of response in KBs
 - Response time
 - Line chart representing response time in ms

Monitoring Mule applications using the Runtime Manager dashboard



Monitoring applications - Dashboard



- Both CloudHub workers and customer-hosted Mule runtimes will report statistics in the Dashboard
- The dashboard shows graphs for three separate metrics for deployed Mule applications and the systems where they're deployed to
 - **Mule Messages**
 - **CPU usage, as a percentage of the capacity**
 - **Memory usage**
- All graphs can be viewed at **different time scales** by selecting the desired time interval on the top-right corner

Monitoring multiple Mule runtimes - Dashboard



- If your Mule application runs on **multiple workers** at a time, they will be charted as different curves on the same graphs, differentiated by different colors
- If your Mule application runs on a cluster or server group, the aggregated metrics of the entire set of servers included will be charted as a single plot line

Exercise 9-3: Access the Runtime Manager dashboard for a Mule application



- Login to anypoint.mulesoft.com and go to Runtime Manager
- Go to American Flights API version v1
- Look at the Dashboard for the Mule application
- Observe the numbers of metrics available in the dashboard for Mule applications and the servers to which the Mule application is deployed
- Verify the metrics for the Mule application's multiple workers for various time spans

- What are the differences and similarities between the API Manager dashboard and the Runtime Manager dashboard?
- What information is missing in either of these dashboards, and what scenarios would require this missing information?

Configuring alerts in Runtime Manager



- **Alerts** can be configured in Runtime Manager
 - Alerts are set up for applications or servers as source at various severity such as critical, warning and info
 - Different event types are available for different event sources (Mule applications vs. servers)
- These can be triggered by Mule application or server conditions
- Alerts can send notifications to email addresses or to Anypoint Platform users within Runtime Manager
- Custom notifications can be generated by a Mule application using the CloudHub Connector, which can generate custom alerts in Runtime Manager
- Note: These alerts are distinct from API Manager alerts and Monitoring Center alerts

Alert conditions related to Mule applications

CH Mule application alerts

- CPU Usage
- Memory Usage
- Custom Notification alert
- Exceeds event traffic threshold
- Secure data gateway disconnected
- Secure data gateway connected
- Worker not responding
- Deployment success
- Deployment failure

On-prem Mule application alerts

- Number of errors
- Number of mule messages
- Response time
- Application Deployment success
- Application Deployment failure
- Application undeployed

Server alerts

- Server up
- Server disconnected
- New server registered
- Agent's version changed
- Runtime's version changed
- Server deleted
- CPU Usage
- Memory Usage
- Server Load Average
- Server Thread Count

Server Group alerts

- Server added to a Server Group
- Server removed from a Server Group
- Server added to a Server Group
- Server removed from a Server Group
- Server group is up
- Server group is partially up (some servers are not running!)
- Server group is down
- A server group's node came up
- A server group's node went down

Cluster alerts

- Cluster Created
- Cluster Deleted
- Server added to a Cluster
- Server removed from a Cluster
- Cluster is up
- Cluster is down
- A cluster's node came up
- A cluster's node went down
- Cluster presents visibility issues

Monitoring applications - Notification

- Notifications give you the ability to give visibility into business related events inside your application
- Notifications are accessible from the Runtime Manager console
- If the notification is sent after an exception, it attaches the exception.message and exception.stacktrace as custom properties of the notification



- How are alerts defined and what types of alerts are possible?
- What types of alerts are missing, and which scenarios require these missing alert types?
- How can you add new alerts to a Mule application in various runtime planes?
- How are event notifications exchanged and handled?
- How can alert notifications be shared with other monitoring systems?

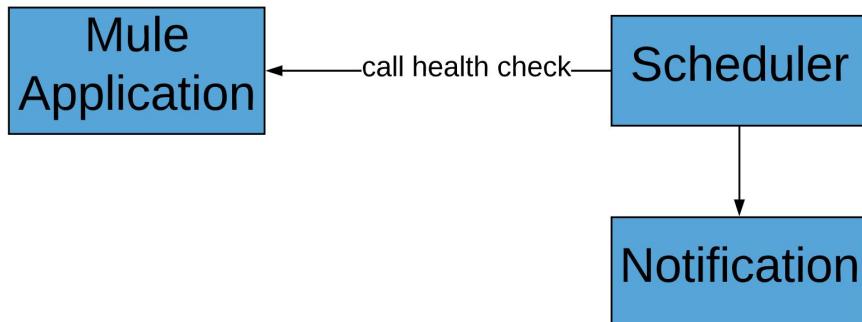
Other Anypoint Platform monitoring and visualization options



Monitoring applications - using custom component



- Any Mule application can define custom health check endpoints, typically accessible via HTTP/S
- An external application or a Functional Monitor can then invoke these health check endpoints at regular intervals



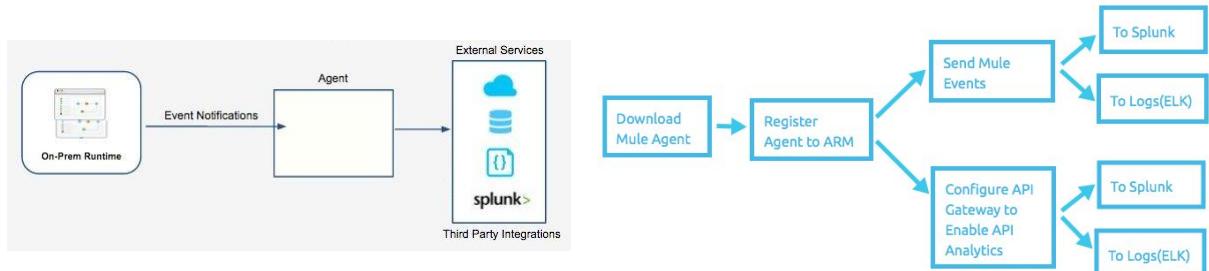
All contents © MuleSoft Inc.

62

Monitoring applications - using Splunk and ELK



- Applications that you deploy on-premises or to your own cloud servers can be integrated into third-party analytics applications
- **The Mule Runtime agent** enables integration and sends analytics/event notifications to your third party analytics tool



All contents © MuleSoft Inc.

63

- Anypoint platform has reporting capability to gather analytics for APIs in CSV and JSON format
- Report provides key details such as
 - Application name
 - API name and version
 - Response time
 - Status code
 - Violated policy
- Runs on demand
- Can download sample report

Monitoring Mule applications using business events and Anypoint Platform Insight



- Mule applications can collect **business events** information as each Mule event transitions through a flow
 - A Mule event's transition across a flow is also called a **transaction**
- Business events are designed to collect key performance indicators (KPIs) and store them in a MuleSoft provided online service
 - Is a convenient alternate way to log KPIs and to troubleshoot issues
- Business events are buffered in the Mule runtime's Runtime Manager agent, then synchronized with the **online** multi-tenant Anypoint Platform Business Events service
- **This process may degrade network or Mule application performance**

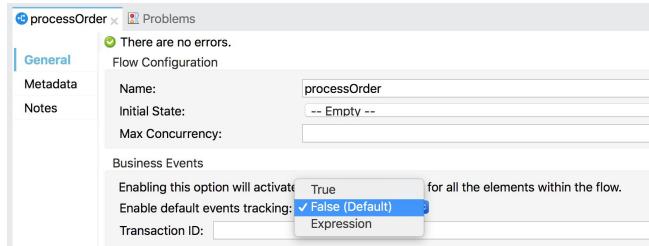
How and when to use business events

- When you want to store information in a MuleSoft provided multi-tenant online storage service
- Usually for coarse-grained integration related KPIs relating to entire interactions rather than specific Mule applications
 - At the end of a long chain of Mule applications, count if there were any errors or not writing to various systems
 - To confirm a performance guarantee, count the number of records in an entire batch job
- Then aggregate KPIs business events to get counts to confirm quality SLAs and generate alerts if needed, or perform other auditing

Tracking default business events



- **Default business events** can be enabled or disabled per flow
 - Default business events track and store every Mule event state through every event processor in the flow
 - Can be used even if default business event tracking is disabled
 - Can also be used to replay a Mule event from the flow's event source
- Some event processors and scopes can also enable or disable business events
 - For example, a Choice router



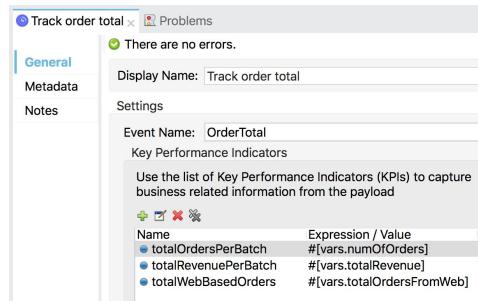
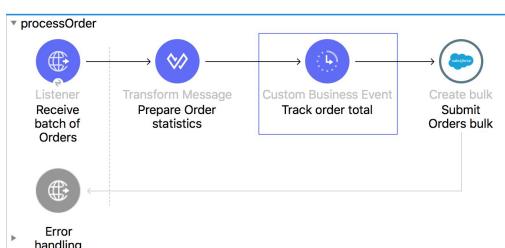
All contents © MuleSoft Inc.

68

Tracking custom business events



- A **Custom Business Event** component can generate additional business events at a particular place in a flow
 - Contains a list of KPIs
 - Can be added anywhere in the flow
 - Is an alternative to using the Logging component so that the Mule application can store KPIs outside the Mule application logs



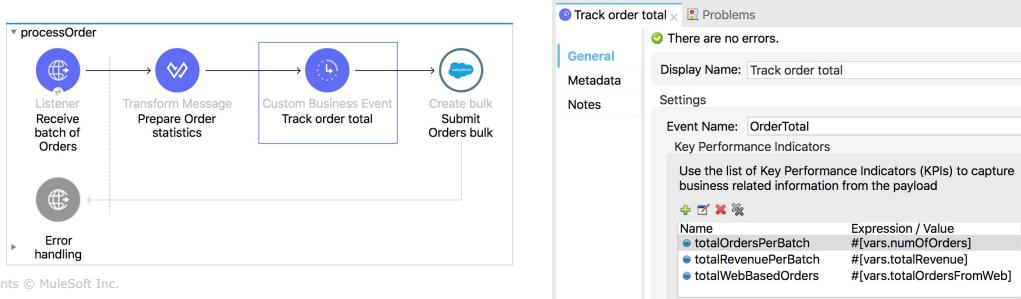
All contents © MuleSoft Inc.

69

What can be stored in a custom business event?



- A custom business event can store the result of any DataWeave expression
- However, the design intent of business events is to only store small sized, usually coarse-grained, KPIs
 - Such as execution times, errors, result completion or failure notification
 - Usually should not include the entire event payload



All contents © MuleSoft Inc.

70

Monitoring a Mule application's business events using Insight



- **Insight** can also be used as a troubleshooting tool that gives you in-depth visibility into business transactions and Mule events
 - Through the inspection of Business Events
- Insight helps to
 - Search transactions
 - Provide information about transactions
 - Status such as success/failure, processing time
 - Find and recover from any errors that occurred during message processing and replay your transactions instantly
- Using Insight, you can monitor business events at runtime
 - To analyze the root cause of failures, isolate performance bottlenecks, and test for compliance to company procedures

All contents © MuleSoft Inc.

71

Enabling default business events for a Mule application



- Enabled in the Runtime Manager Insight tab
 - In Insight it is called Metadata
- For Mule applications deployed to CloudHub, event replay from each flow's event source can also be enabled
 - Every Mule event is then streamed from the Mule runtime to the online business events storage service
 - Should only be used briefly for troubleshooting purposes
- Default business events tracking will be enabled for every flow and component
 - Even if it is currently configured as disabled in the Mule application

The screenshot shows the 'Insight' tab selected in the left sidebar of the Runtime Manager. The main area displays the application file 'process-orders.jar' with its last update time and app URL. Below this, the 'Event Replay' section is shown with three options: 'Disabled' (radio button), 'Metadata' (radio button), and 'Metadata and Replay' (radio button, which is selected). A note under 'Metadata and Replay' states: 'Store message metadata and additional information needed to allow transaction replay.' At the bottom right is a blue 'Apply Changes' button.

Insight performance impact and limitations



- Enabling Insight has a **large performance impact** when processing application data
 - Typically Insight should not be enabled for extended periods (or ever) in production environments
- Replay only works for flows that have inbound endpoints that are deployed to CloudHub workers
- Batch processing is not supported
- The Anypoint Platform - Private Cloud Edition doesn't currently support Anypoint Insight

- What use cases would require using Business Events and Anypoint Insight?
- What are the tradeoffs of using Insight in a production environment?
- What are some alternatives to using Business Events, and what are the tradeoffs?
- Can Business Events be exchanged with another log management system such as Splunk or ELK?
- Can Business Events be written to a log file?

Summary

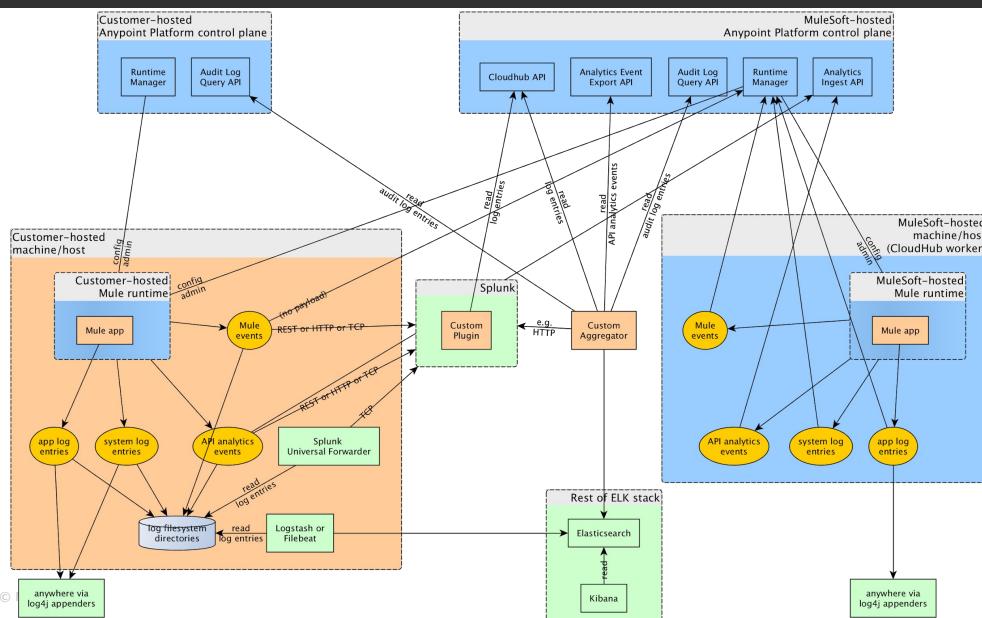


Summary



- Logger processor logs the entire Mule event, including all attributes, as well as flow variables except content of payload
- By default, logging in Mule is done asynchronously
- Anypoint platform provides Anypoint Monitoring, Analytics, Insights, Dashboards and Flow metrics for analyzing and monitoring applications/servers for performance and debugging
- Anypoint Monitoring is next generation tool from MuleSoft for analyzing and monitoring applications/servers for performance and debugging

Overview of log integrations



- CloudHub Custom Log Appender
 - <https://docs.mulesoft.com/runtime-manager/custom-log-appender>
- Splunk (Universal Forwarder):
 - <http://docs.splunk.com/Documentation/Forwarder/7.1.1/Forwarder/Abouttheuniversalforwarder>
- Log4j specific-system **appenders**
 - Splunk Logging Java:
http://dev.splunk.com/view/splunk-logging-java/SP-CAAAE3P#_Maven
 - TCP/Socket appender:
<http://dev.splunk.com/view/splunk-logging-java/SP-CAAAE3R>



Module 10: Designing an Efficient and Automated Software Development Lifecycle



Goal



The screenshot shows the Anypoint Platform interface with the following elements:

- Left Sidebar:** A file tree showing configuration files: aesEncryptionKey.jceks, clientStore.jks, config-http.yaml, mule-property (selected), and serverStore.jks.
- Middle Column:** A list of environments: dev (selected), prod, and stage.
- Right Column:** A file named property-management.yaml.
- Bottom Panel:** A terminal window titled "Anypoint Platform" showing the command-line interface (CLI) v2.1.0. The output includes:
 - Connecting to the Anypoint Platform...
 - Connected to OnPrem01 Group07 - Production
 - > account environment list
 - A table listing environments:

Name	Id	Sandbox
Production	9d6e016c-abc8-4e6c-a78b-166f410adaef	N
Sandbox	85d3fb4-4163-4c50-8acb-a4344dfa033e	Y

> account environment create --sandbox QA

All contents © MuleSoft Inc.

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



- Manage property files for Mule applications across different environments
- Manage Anypoint Platform environments for Mule application deployments
- Design testing strategies for Mule applications
- Implement continuous integration and continuous delivery (CI/CD) for an organization
- Automate deployment and management with Anypoint Platform

Managing property files for different environments



Using properties to help Mule applications evolve



- Mule applications may need to use different configurations between Mule runtimes
 - Examples
 - To avoid TCP bind errors
 - To configure values for a SDLC environment
 - To configure values for a region
 - Runtime Manager provides a UI to configure Mule application properties
 - Mule application properties can also be configured without Runtime Manager
- These phases/regions/environments have differences
 - They use different data (Live vs. non-live data)
 - Different databases or other systems of record, with different restrictions
 - They often use different credentials to access data
 - Different values
 - Non-encrypted vs encrypted



All contents © MuleSoft Inc.

5

Configuring Mule application properties



- Properties that might change include
 - Connector properties
 - Location information like hostnames and ports
 - Performance tuning values like connection pools, thread pools, and timeout values
 - Security properties
 - Names of environments, files, and resources
 - External locations for files and other resources
 - Performance tuning properties
 - Connection properties such as reconnection limits and connection pools
 - Other tuning properties like time-out values and polling frequencies
 - Other application properties like JMS message configurations



All contents © MuleSoft Inc.

6

Managing Mule application and system level properties using a Mule Runtime



- Mule application level properties
 - Only visible to the deployed Mule application
 - Stored in the Mule application deployable JAR
- System level properties
 - Set as JVM system variables
 - Shared by every Mule application deployed into the Mule runtime
 - Can be set in the wrapper.conf file or from the command line when starting the Mule runtime
 - Will override Mule application level properties set by the Mule application's deployable JAR

<https://support.mulesoft.com/s/article/How-can-I-set-Mule-and-Java-system-properties-at-startup>

All contents © MuleSoft Inc.

7

Architects need to design a way to externalize Mule application properties



- Configuration values that change should be externalized by developers into configuration files
 - They should not be hard-coded inside Mule applications
- External configuration files make Mule applications easier to upgrade and migrate
- External configuration management systems can also be used, but require more work to set up and manage
 - Typically requires development of a custom integration with the Mule runtime

All contents © MuleSoft Inc.

9

- Create a separate configuration file for some/all environment

```
dev-prop.yaml           prod-prop.yaml
sqldb:
    user: "mule"
    password: "mulemax"
sqldb:
    user: "mule"
    password: "! [gCs37bbR6NDrTrABIxHhOA==]"
```

- Details in the Anypoint Platform Operations: Customer-Hosted Runtimes course

Managing environments across an organization



- The **Organization Owner** is the Anypoint Platform user that first signs up for an Anypoint Platform account
 - It is the user that pays for the Anypoint Platform account
 - This is not a role but an identifier for this single user
 - Inherits the **Organization Administrator role** by **default**
- This upper level organization has a client id and client secret used to secure some Anypoint Platform features
- The upper level organization has either one external idP for identity management or none
 - In which case Anypoint Platform performs identify management

Managing business groups, users, roles, and environments

- Each organization can contain child **business groups**
- The organization and its business groups can each define different roles, environments, and users
- Roles define and manage user permissions across Anypoint Platform, for specific environments within a particular business group
 - Some predefined roles are created for every business group
 - Custom roles can also collect a different set of permissions

- Business groups provide complete **isolation** of resources
- vCores are **assigned** to a specific business groups
 - Makes those vCores only available to the business group and unavailable to the parent organization or business group
- Each business group has its **own** environments
- Each business group has a **separate** client id and client secret
- Deleting a business group is NOT recoverable as all resources get deleted

How to delegate administration of an organization and its child business groups

- The **organization owner** has **all permissions** for its organization and child business groups, independent of any roles
- Other users can be added to the Administrator role of the upper organization, and then also have **all permissions**
- Each **business group** also has an **owner** with full administrator role privileges to that business group and its child business groups
 - Has full permission to create, change, or remove any role, user, or child business group
 - But **not to parent** business groups

- Anypoint Platform supports the following **types** of environments
 - **Production** quality environments
 - Where you can deploy Mule applications and APIs publicly
 - When you create a new Anypoint Platform account, by default it contains one production environment
 - **Sandbox** quality environments
 - Provide useful environments for development and testing
 - By default, Anypoint Platform accounts are created with one sandbox environment
 - **Design** quality environments
 - Enables you to test and run Mule applications at design time
 - By default, Anypoint Platform accounts are created with one design environment

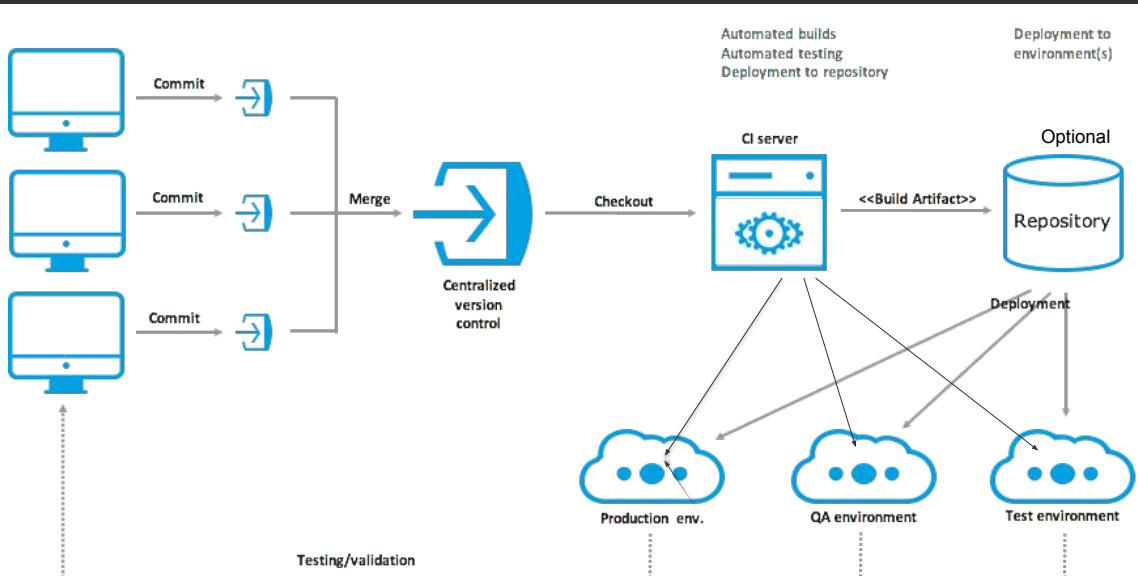
- Anypoint Platform licenses the number of VPCs available to an organization
- An Anypoint Platform VPC can be configured to span across multiple environments and child business groups under the current business group
- Each VPC is assigned to a business group

Reference: Anypoint Platform Operations: CloudHub

Implementing continuous integration and continuous delivery (CI/CD)



Automation using CI CD



Implementing CI CD on Anypoint platform



- Anypoint Platform supports continuous integration and continuous delivery using industry standard tools
 - The **Mule Maven plugin** can automate building, packaging, and deployment of Mule applications from source projects
 - The **MUnit Maven plugin** can automate test execution, and ties in with the Mule Maven plugin

Implementing CI CD on Anypoint platform using Anypoint Exchange



- **Anypoint Exchange** is a central repository for various types of assets
 - So they can be discovered and reused to build your integration projects
- Stores and shares enterprise assets
 - RAML fragments
 - Custom Connectors
 - Project templates and examples
- Is a Maven compatible artifact repository
 - Is not intended as a replacement for a full version control system

- Anypoint Studio includes Maven support
- Anypoint Studio can generate and manage the Mule application's pom.xml file automatically
 - For example, adding a Salesforce connector imports a compatible SFDC library
- All Anypoint Studio Mule projects are automatically **mavenized**
- The Maven plugin supports Mule domain and Mule application projects

How the Maven plugins supports Mule application lifecycle phases

- Together, the Mule and MUnit Maven plugins implement MuleSoft behavior in various Maven lifecycle phases
 - compile - Compile the Mule source code of the project
 - test - Runs MUnit tests associated with project
 - package - Packages the project into a Mule deployable jar
 - install - Sends the distributable to \$MULE_HOME, a local repository
 - deploy – Sends the distributable to a remote repository
- Triggering a later phase in the lifecycle triggers all phases before it

- To use Maven to deploy an application to a server, use the mule-maven-plugin
- Supports deployments to
 - CloudHub
 - Anypoint Runtime Manager
 - Standalone runtime
 - Mule Agent
- For more information on deployment for specific scenario
 - <https://docs.mulesoft.com/mule-runtime/4.1/mmp-deployment-concept>

How to implement shared resources in Mule project

- Every Mule application can be assigned to one Mule domain in the Mule application's configurations pom.xml file
- The Mule domain project is referenced in each of the Mule applications that are meant to use these share resources
- Mule applications can be associated with **only one** domain at a time
- Maven supports creation of Mule domain project

- Mule domain projects can only be manually deployed to customer-hosted Mule runtimes
 - Runtime Manager **does not support** deployment or management of Mule domains, but you can deploy and manage Mule applications to an existing Mule domain already running in the Mule runtime(s)
- Runtime Manager does not support deployment of domain project

Resource to understand how to design and implement CI/CD for Mule applications

- The student files have documents to help with CI/CD
- Reference: Anypoint Platform Development: Advanced

- When are Mule domains useful, and what are the tradeoffs of coupling Mule applications with a Mule domain?
- Why are Mule domains not needed in CloudHub?
- When and why would Mule domains be useful in Runtime Fabric deployments?
- How are these shared resources implemented and how do they behave between Mule applications in a Mule domain?
 - HTTP Listener
 - HTTP Request
 - VM Listener
 - Object Store connector
 - File Listener (On New or Updated File operation)
 - Database Listener (On Table Row operation)

Automating Anypoint Platform



- Anypoint Platform and the Mule Maven plugin both provide multiple option for automation
 - Anypoint Command Line Interface (CLI)
 - Anypoint Platform APIs
- Automate Anypoint Platform administrative activities with the
 - Anypoint-CLI command-line tool
 - Combines REST API steps into easier to use commands
 - Anypoint Platform APIs
 - First generate an access token or use OAuth2
 - Use the access token in other REST calls
 - Set the environment id or organization id as needed

Automation using Anypoint Command Line Interface (CLI)

- Is a Node.js based application
- Used to access Anypoint Platform APIs
- Anypoint CLI commands simplify common use cases
 - Authentication using using username/password not secure access token
 - Set organization, environment using name rather than the ids
 - Can look up resources using name rather than id
- Runs in interactive or non-interactive mode
 - In interactive mode, the user type into command line interface to perform automation of tasks
 - In non-interactive mode, the user create script file to perform automation of tasks and is preferred for repetitive tasks
- Reference
 - <https://docs.mulesoft.com/runtime-manager/anypoint-platform-cli>

- Anypoint Platform APIs than can help orchestrate API based deployment and management of CI/CD automation
 - MuleSoft Developer Portal
 - <https://anypoint.mulesoft.com/exchange/portals/anypoint-platform/>
 - Access Management API
 - <https://anypoint.mulesoft.com/exchange/portals/anypoint-platform/f1e97bc6-315a-4490-82a7-23abe036327a.anypoint-platform/access-management-api/>
 - CloudHub API
 - The CloudHub Public API enables you to access application management services for applications deployed to CloudHub
 - <https://anypoint.mulesoft.com/exchange/portals/anypoint-platform/f1e97bc6-315a-4490-82a7-23abe036327a.anypoint-platform/cloudbus-api/>

Automation using Anypoint platform APIs (cont.)

- Other Anypoint Platform APIs than can help orchestrate API based deployment and management of CI/CD automation
 - API Manager API
 - The API Manager API enables you to manage an API by applying policies, setting SLAs, configuring alerts for your API instances, and promoting API instances
 - <https://anypoint.mulesoft.com/exchange/portals/anypoint-platform/f1e97bc6-315a-4490-82a7-23abe036327a.anypoint-platform/api-manager-api/>
 - API platform v2
 - The API Platform API exposes the management capabilities of the Anypoint Platform for APIs, enabling them to be used by external sites
 - <https://anypoint.mulesoft.com/exchange/portals/anypoint-platform/f1e97bc6-315a-4490-82a7-23abe036327a.anypoint-platform/api-platform-api/>
 - Anypoint Exchange API and Proxies API
 - Please refer <https://docs.mulesoft.com/release-notes/upgrade>

- When and how would you use the Anypoint CLI?
- When would you use the Anypoint Platform REST APIs instead, or in conjunction with the Anypoint CLI?
- What are some ways you can build automated scripts with Anypoint CLI and the Anypoint Platform REST APIs?

Summary



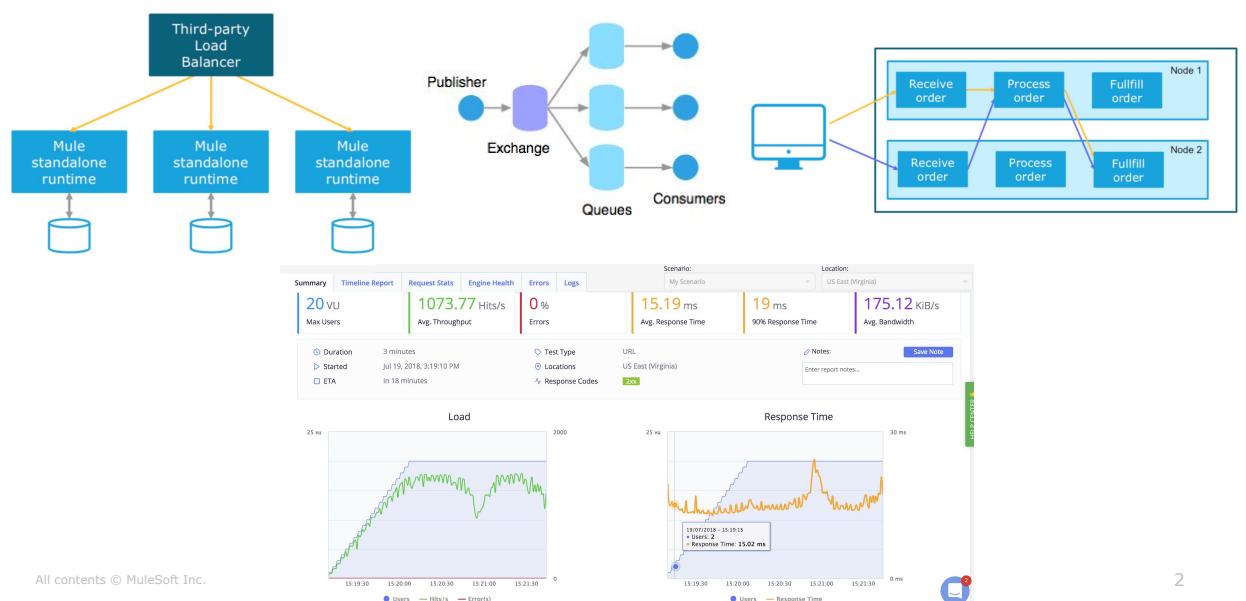
- DevOps requires having a strategy to manage properties for different environments
- MUnit promotes test driven development for an organization
- Simplifying CI and CD pipelines is also key for organizational success
- MUnit helps developers to unit test APIs throughout a SDLC



Part 3: Strategies to Meet Non-Functional Requirements



Goal



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



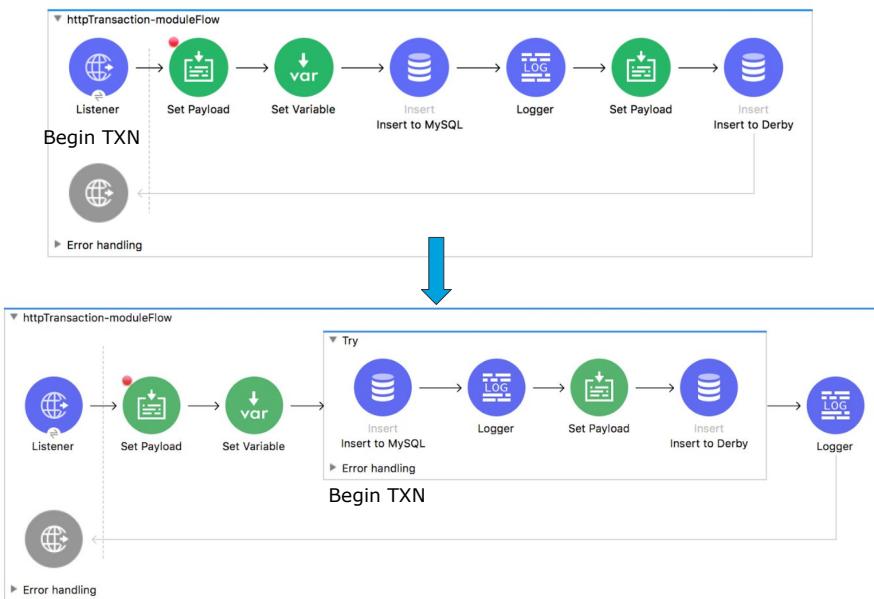
- Decide when and how to design for transactional requirements
- Clarify and balance reliability, high availability, and performance goals
- Design to meet agreed reliability, high availability, and performance goals
- Design secure Mule applications, their network communications, and their deployments
- Summarize the course with a completed architecture



Module 11: Designing Transaction Management in Mule Applications



Goal



5

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

- Identify why and when transactions are supported in Mule applications
- Understand resources that participate in transactions in Mule applications
- Demarcate transaction boundaries in Mule applications
- Choose the transaction type based on the participating resources
- Manage a transaction using the Saga pattern

Identifying why and when transactions should be used in Mule applications



What is a transaction?



- In its narrow technical definition, a **transaction** is a grouping of operations that are guaranteed to all complete, or none at all
 - This is the definition often used in computer science
 - Traditionally were implemented to meet ACID guarantees
 - Atomicity
 - Consistency
 - Isolation
 - Durability
- A **business transaction** is a much more general term that comprises potentially any kind of finite activity with business meaning
- A **financial transaction** is a transfer of funds, either in a short time or over an extended period of time
 - For example, paying a mortgage over 30 years is a financial transaction

How traditional two-phase commit protocols are applied to distributed systems



- Databases and related systems **traditionally** use two-phase commit protocols to manage transactions
 - Transactions progress through a prepare and a commit phase across all participants of the two-phase commit transaction
 - The hope is to guarantee all the separate operations complete all at once, or none of them complete and they are all rolled back
 - They maintain the ACID properties of a transaction
 - They require heavy-weight **transaction manager** services
- **Global transactions** are transactions that involve multiple separate transactional resources (such as databases)
 - A **transaction manager** is used to **atomically** coordinate and rollup the individual transaction of each resource

How XA-capable transaction resources participate in global/distributed transactions



- Resource managers participating in global transactions, using two-phase commit, often implement the **XA interface**, so they can be managed by a transaction manager
 - Allows separate protocols and systems to be combined into one global transaction
- The XA interface specifies communication between a **transaction manager (TM)** and a **resource manager (RM)**
 - Each RM must respond to requests from the TM to **prepare**, **rollback**, or **commit** an operation in its managed resource

Problems with two-phase commit protocols applied to modern large distributed systems



- Requires remote communication to each participating system
- Requires each participating system to support two-phase commit
- Locks (parts of) each participating system for the duration of the transaction
- Communication protocol grows linearly (O(N)) with the number of nodes

Alternatives to two-phase commit protocols for distributed transactions



- Large **distributed transactions systems** are available to improve upon two-phase commit protocols
 - Such as Google Spanner
- **Eventual consistency** is often used in large distributed systems, but this compromises the **consistency** goals of 2-phase commit transactions
- **Sagas** are another alternative to global distributed transactions
 - Requires explicitly coding and managing compensating transactions
- All these alternatives can be characterized through the **CAP theorem**
 - Can only have two of Consistency, Availability, and Partition Tolerance
 - https://en.wikipedia.org/wiki/CAP_theorem

Managing transactions using the Saga pattern



Managing transactions using the Saga pattern



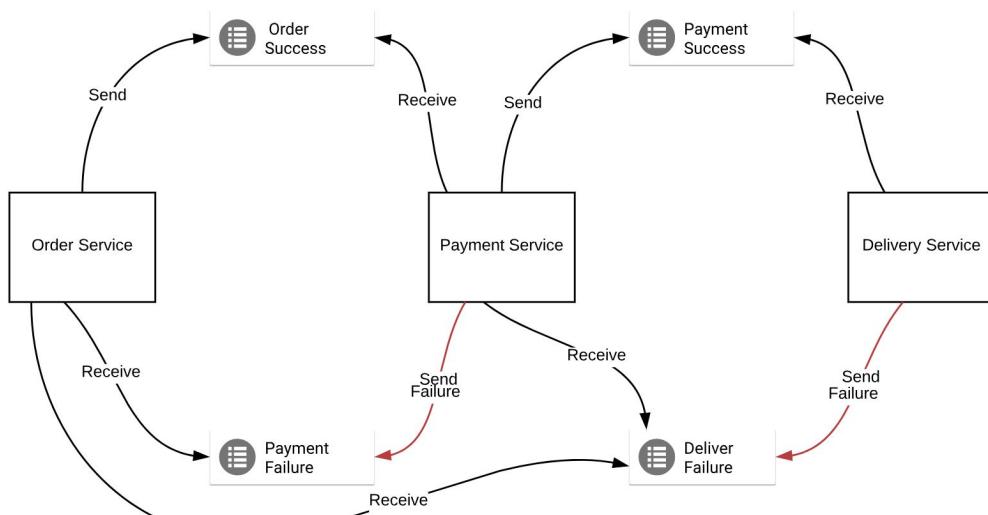
- The Saga pattern is an alternative to XA when participating resources are not XA capable or XA is undesired
 - For instance, API invocations
 - A way to achieve reliability that does not have ACID guarantees
- Features of the Saga pattern include
 - Maintains consistency (but not atomicity) across multiple services without requiring a **transaction manager system**
 - Uses a **sequence** of local TXs
 - Failure of a local TX executes a **series of compensating transactions** to undo the previous successful TXs
 - This leads to a more complex programming model compared with XA resources

Some ways to implement Saga patterns

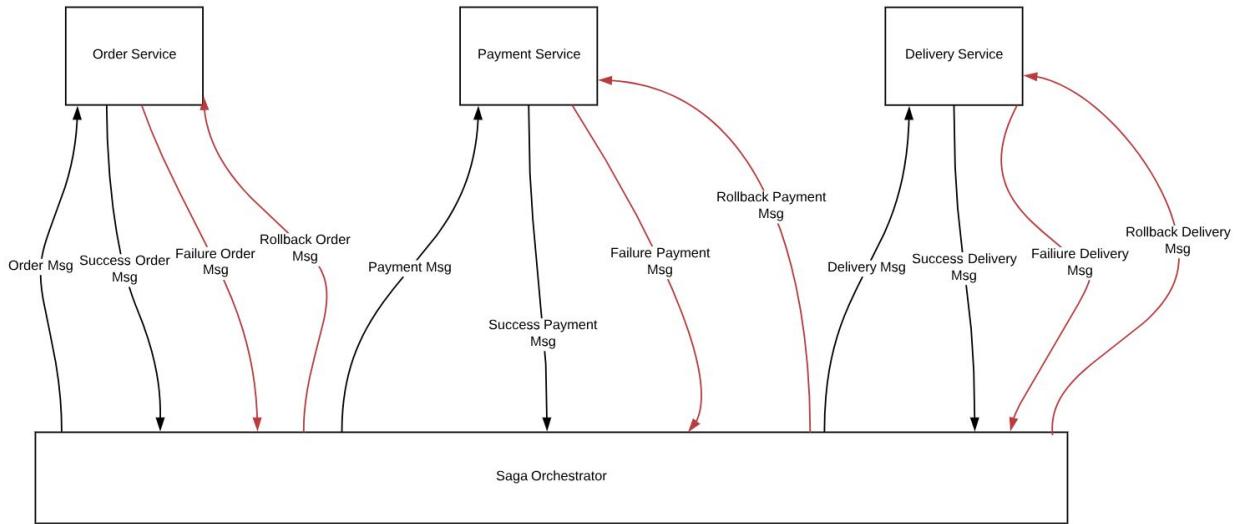


- Event/Choreography pattern
 - The service executes the transaction (**TX**) and publishes an event to be used by the next service
- Command/Orchestration pattern
 - A **Saga orchestrator** communicates with each service using a command/reply style
 - The orchestrator logic can be modeled as a state machine
 - Similar to an XA pattern, the orchestrator coordinates transactions
 - But does not require an XA-capable transaction manager nor resource managers
 - So this works with API invocations, messages, etc.

Example: Saga Event/Choreography pattern



Example: Saga Command/Orchestration pattern



Tradeoffs of Event/Choreography vs Command/Orchestration patterns



	Event/Choreography	Command/Orchestration
Characteristics	<ul style="list-style-type: none">• Listens for successful event from previous step and failure events from subsequent steps	<ul style="list-style-type: none">• Listens for success and failure messages from the Saga orchestrator
Pros	<ul style="list-style-type: none">• Use when coordination across domains of control/visibility is required	<ul style="list-style-type: none">• Use to manage business processes that are inside one organisation that you have control over• Centralizes orchestration of TX• Complexity grows linearly when steps are added
Cons	<ul style="list-style-type: none">• Adding and deleting steps in a TX is more difficult	<ul style="list-style-type: none">• Single point of failure

Exercise 11-1: Identify when and why a Mule application should support transactions



- A Mule application makes multiple related modifications, and at least one fails
- Answer these questions
 - How can the consistency of the entire system be guaranteed?
 - What does consistency mean?
 - Discuss the different aspects of this scenario, including edge cases and performance considerations

Exercise solution



- Maintain data integrity with transactions or Sagas
- Characteristics
 - Data integrity
 - Atomicity
 - Consistency
 - Isolation
 - Durability
 - Performance
 - Can decide the size of the unit of work that is committed in one transaction
 - Distributed transactions are slow because of logging and network communication
 - Performance overhead of locking multiple resources in a transaction

- Sagas enable an application to maintain data consistency across multiple services without using distributed transactions:
 - Performance
 - Complex programming model
 - Must design compensating transactions that explicitly undo changes made earlier in a Saga

Managing transactions in Mule applications



- Mule supports
 - Single-resource (local) TX
 - For example, a series of operations to one database connection
 - Global (XA) TX
 - A series of operations between one or more (typically two) different (XA compatible) systems, such as two different databases, or a database and a JMS server
- Supported connectors must be configured to use transactions
- All message processing done on a single thread
 - Bypasses typical reactive thread usage
- XA transactions need an XA-capable transaction manager

Connector operations that support transactions

- JMS
 - Publish
 - Consume
- VM
 - Publish
 - Consume
- Database
 - All operations

Demarcating the beginning of a transaction



- Begin a transaction through either
 - A Try scope
 - A transactional connector acting as an event source
 - Database, JMS, or VM Listeners

Demarcating the end of a transaction

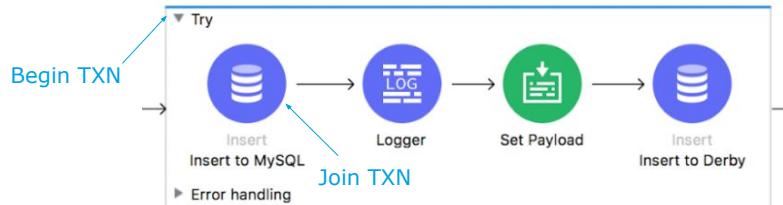


- Committed automatically at end of a
 - Flow
 - Try scope
 - On Error Continue scope
- Rollback transaction
 - After a failure occurs in a transaction scope, but only if the error is not handled in an On Error Continue scope
 - On Error Propagate scope
 - By throwing an error in a flow or in a Try scope using <raise-error>
 - Note: This component is not currently available in an Anypoint Studio palette, it must be typed into the Mule XML configuration file

Demarcate transaction boundaries



```
<!-- Local Transaction Begins -->
<try transactionalAction="ALWAYS_BEGIN">
    <db:insert config-ref="Derby_Database_Config"
transactionalAction="ALWAYS_JOIN"/>
    <!-- Rollback Transaction in case of error -->
    <error-handler>
        <on-error-propagate>
            <raise-error type="ANY" description="#[Rollback]"/>
        </on-error-propagate>
    </error-handler>
</try>
<!-- Transaction Ends -->
```



Transaction types in Mule applications



Transaction Type	Characteristics	Number of resources	Available resources	Performance
Single Resource/ Local	<ul style="list-style-type: none">Receive and/or send message to only one resourcePerform database operation to only one database resource	1	JMS VM JDBC	relative to XA performs better
XA	<ul style="list-style-type: none">Receive and/or send message to one or more resourcesPerform database operation to one or more transactional resourcesConnectors must be XA enabledUses two phase commit	>=1	JMS VM JDBC	relative to local slower, but ACID across all resources

- The TX log stored in \$MULE_HOME/.mule/TX-log
- By default, this file will contain up to 50000 records
- For performance, the TX log size can be optimized in MB using
`<configuration maxQueueTransactionFilesSize="150" />`
- Logs are not human readable and are available during the lifetime of the TX
- The TX log contains begin, commit, rollback, and isolation details for each TX
- A TX manager has its own TX log file - BTM, XA, and Local
- The TX log enables the Mule runtime to rollback or restore a TX in case of a hardware failure or Mule application failure.

Using the default Bitronix XA transaction manager in Mule applications

- **Bitronix** is available as the XA transaction manager for Mule applications
- To use Bitronix, declare it as a global configuration element in the Mule application
`<bti:transaction-manager />`
- Each **Mule runtime** can have only one instance of a Bitronix transaction manager, which is shared by all Mule applications
- For customer-hosted deployments, define the XA transaction manager in a Mule domain
 - Then share this global element among all Mule applications in the Mule runtime

Setting a transaction timeout for the Bitronix transaction manager



- Set the transaction timeout either
 - In wrapper.conf
 - In CloudHub in the Properties tab of the Mule application deployment
- The default is 60 secs

```
mule.bitronix.transactiontimeout = 120
```

Participating in transactions (TX)



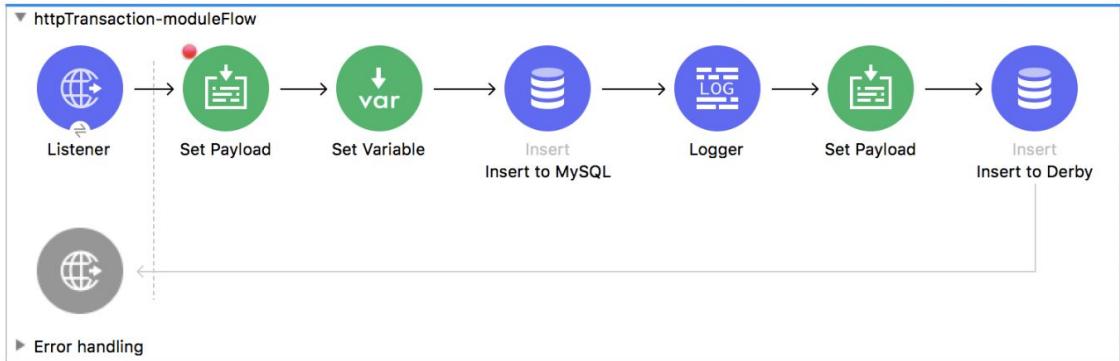
- Defining transactions behaviour on the connector operation

Action	Processor Supports	Description
None	JMS, VM and DB Connectors (listeners only)	Never participate in TX Ignore any ongoing TX
INDIFFERENT	Try scope	Never participate in TX Ignore any ongoing TX
ALWAYS_BEGIN	JMS, VM and DB Connectors (listeners only) and Try scope	Always start new TX Throw exception if TX ongoing

Exercise 11-2: Define a transaction



- For a specific flow, identify transactional resource(s), transactional types, transactional boundaries and their demarcation



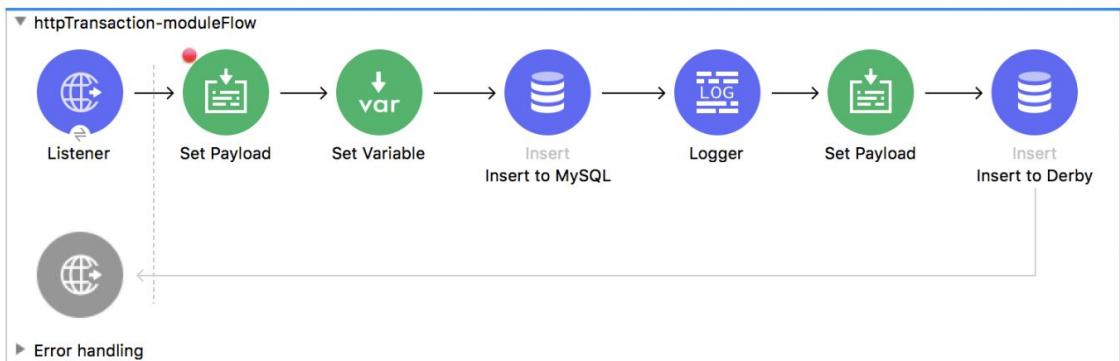
All contents © MuleSoft Inc.

36

Exercise steps



- Identify transaction resources
- Transactional resource(s) are DB, VM and JMS, identify in flow TX resources



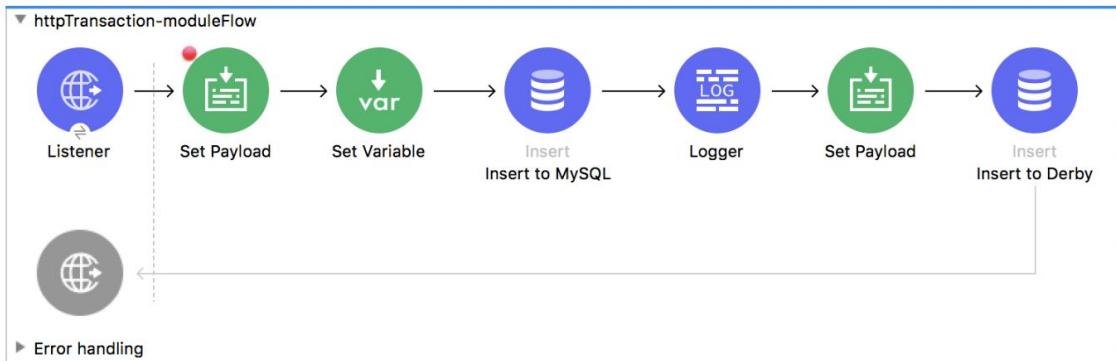
All contents © MuleSoft Inc.

37

Exercise steps



- Define TX boundary and TX type
 - Should we use local/single resource or XA TX
 - Should TX start with listener or have transaction in Catch scope



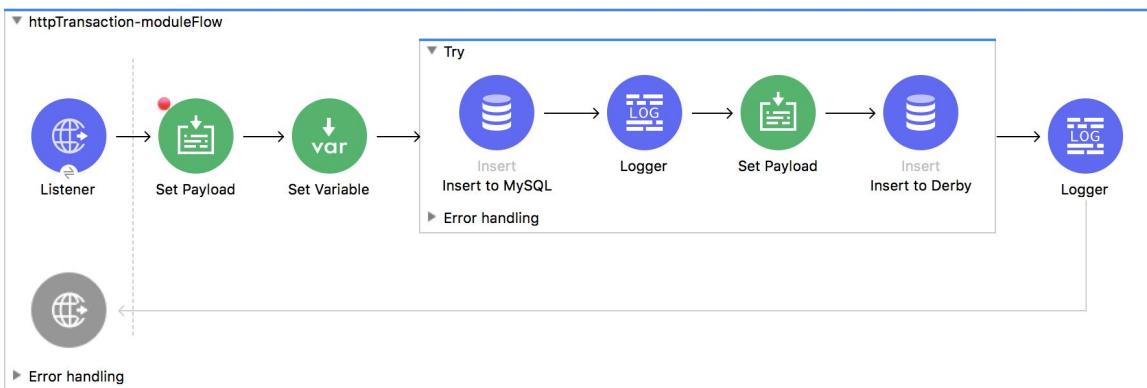
All contents © MuleSoft Inc.

38

Exercise solution



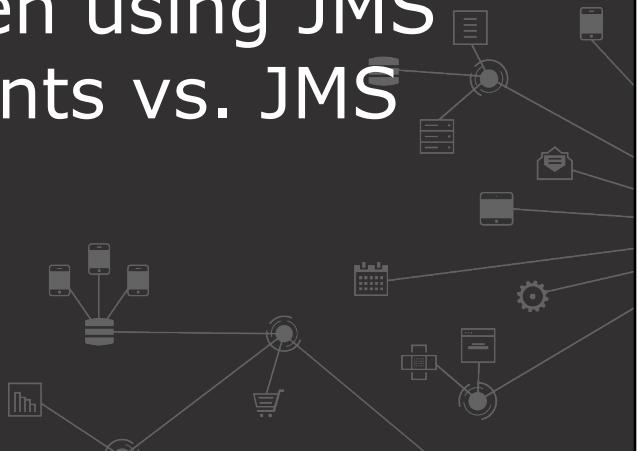
- Define transactions in a Mule application
- Use a Try scope to handle rollback exceptions
- Define



All contents © MuleSoft Inc.

39

Deciding between using JMS acknowledgements vs. JMS transactions



JMS acknowledgement in Mule applications



- Receipt of JMS messages must be acknowledged
 - Assuming JMS transactions are not used
- Unacknowledged messages are **redelivered**
- JMS Recover Session **redelivers** all the consumed messages that had not been acknowledged before this recover
- **JMS Recover session** is used inside Error Handler to redeliver messages
- Acknowledgement mode set on JMS listener
 - auto, manual, immediate, dups_ok

- AUTO (default)
 - Automatic acknowledgement at end of flow or On Error Continue scope
- IMMEDIATE
 - Same as NONE in Mule 3
 - Automatic acknowledgement upon receipt
 - Not redelivered if processing fails afterwards
- MANUAL
 - Use jms:ack operation
- DUPS_OK
 - Similar to AUTO
 - Optimization: acknowledges messages lazily
 - Typically after delay or in bulk
 - Application must handle duplicate message delivery

- **JMS transaction (TX)** means a local transaction involving a JMS broker
- **JMS acknowledgments** are non-transactional messages defined by the JMS protocol for a client to acknowledge receipt of a message to the JMS broker
- **JMS acknowledgements** and **JMS TXs** are mutually exclusive
- **JMS TX**
 - Is analogous to other transactions
 - Applies to sent and/or received messages
 - Can bundle one or several sends/receives in a TX
- **JMS acknowledgments (ACK)**
 - Are specific to JMS brokers
 - Only apply to confirming receipt of received messages
 - Performance and granularity can be tuned

- Use JMS ack if
 - Acknowledgment should occur eventually, perhaps asynchronously
 - The performance of the message receipt is paramount
 - The message processing is idempotent
 - For the choreography portion of the SAGA pattern
- Use JMS transactions
 - For all other times in the integration you want to perform an atomic unit of work
 - When the unit of work comprises more than the receipt of a single message
 - To simplify and unify the programming model (begin/commit/rollback)

Summary

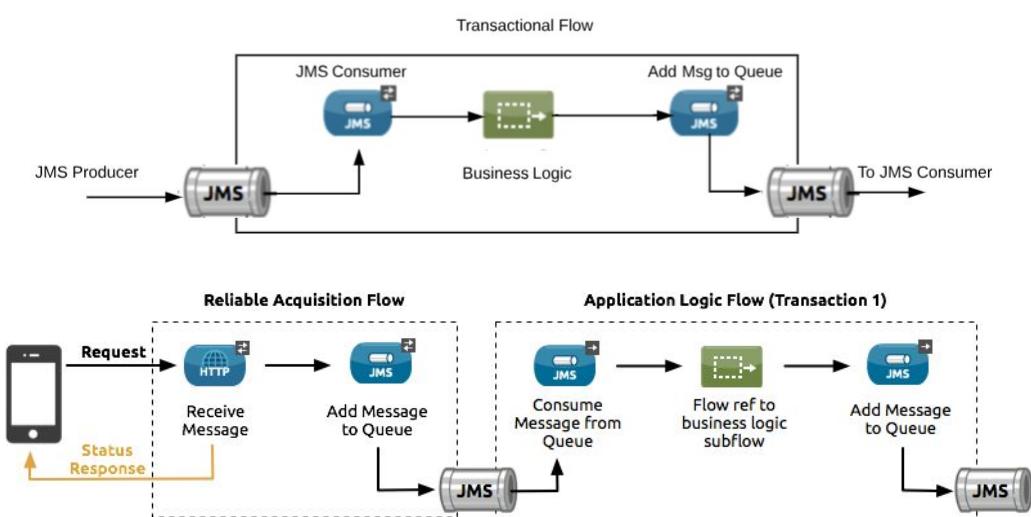


- Mule applications can manage TXs
- Demarcate a TX with connectors supporting TX, or in a Try scope
- Can be local or XA based on the transactional connectors involved
- Begin a TX by selecting an appropriate action (ALWAYS_JOIN etc)
- A TX is committed automatically in the absence of an exception
- Rollback is possible in case of failure within TX boundary
- JMS acknowledgement is an alternative to TX and provides various service levels for message acknowledgement
- The saga pattern provides a pseudo-distributed TX for resources such as REST services

Module 12: Designing for Reliability Goals



Goal



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



- Distinguish between competing non-functional requirements
- Clarify and validate reliability goals for a scenario
- Design Mule applications and their deployments to meet reliability goals
- Identify reliability pattern for mule application and their deployments

Balancing tradeoffs to meet non-functional requirements



Identifying non-functional requirements



- Applications can be designed and tuned for various opposing goals
 - High availability
 - Reliability
 - Performance
 - Fast response time/low latency
 - High throughput
 - Capacity (number of concurrently processed messages)
 - Security
- Performance requirements, service level agreements (SLAs), and other business goals should be clearly agreed upon and documented
 - They are often opposing goals

Considering tradeoffs to meet opposing non-functional requirements



- Cost
- Reliability
 - Business implications of duplicate message processing vs. lost messages
- Competing performance SLAs
 - Fast response time/low latency
 - High throughput
 - Capacity (number of concurrently processed messages)
- Transactional exactly once requirements
 - Latency and cost vs. business risk

Defining and achieving reliability goals



Defining and achieving reliability goals



- **Reliability** aspires to have zero message/data loss after a Mule application stops or crashes
- Various reliability patterns can be implemented to achieve reliability goals for synchronous and asynchronous flows
- Reliability in Mule applications can be achieved using
 - Until Successful scope
 - Reconnection strategies
 - Redelivery policy
 - NS:RETRY_EXHAUSTED exception scope
 - Transactions (covered in module 11 - Transaction)

Achieving reliability goals with Mule components



Achieving reliability using an Until Successful scope



- The **Until Successful** scope repeatedly triggers the scope's components (including flow references) **until they all succeed or until a maximum number of retries is exceeded**
- The scope provides option to control the max number of retries and the interval between retries
- The scope can execute any sequence of processors that may fail for whatever reason and may succeed upon retry

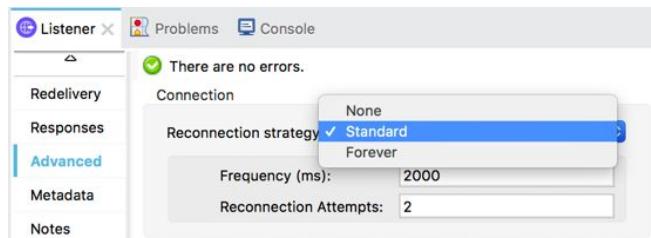
```
<until-successful maxRetries="5"  
    millisBetweenRetries="1000">  
    <!-- One or more processors here -->  
</until-successful>
```



Achieving reliability reconnection strategies



- **System errors** are thrown when a connection to an external system (DB, message broker, etc) fails
- To retry after connection failures, Mule connectors can set a **reconnection strategy**
 - Set for a connector (in the Global Elements Properties) or for a specific connector operation (in the Properties view)



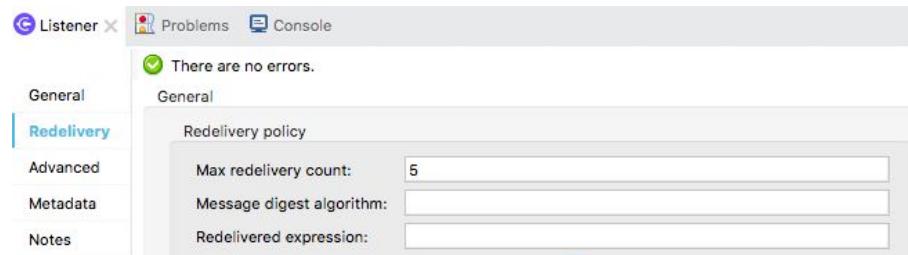
All contents © MuleSoft Inc.

11

Achieving reliability using a redelivery policy



- A **redelivery policy** is configured on inbound connectors, such as the JMS connector, to specify the number of redeliveries before discarding the message
 - 0 means no redelivery
 - -1 means infinite redeliveries



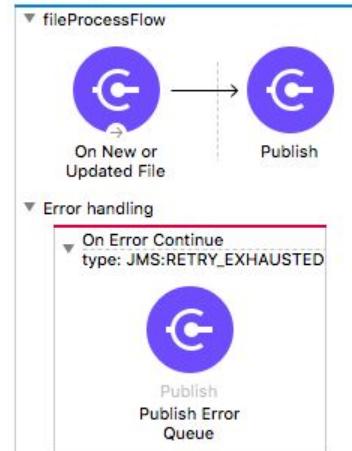
All contents © MuleSoft Inc.

13

Achieving reliability using MuleSoft components - RETRY_EXHAUSTED exception scope



- Before discarding the message after the number of redeliveries attempted, the connectors raises an exception of type RETRY_EXHAUSTED
- An error scope can handle the RETRY_EXHAUSTED error with logic required to handle the error, so the event is not lost



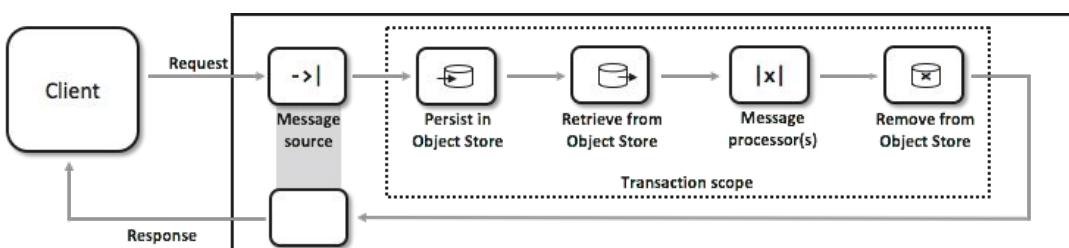
All contents © MuleSoft Inc.

14

Achieving reliability for transactional systems



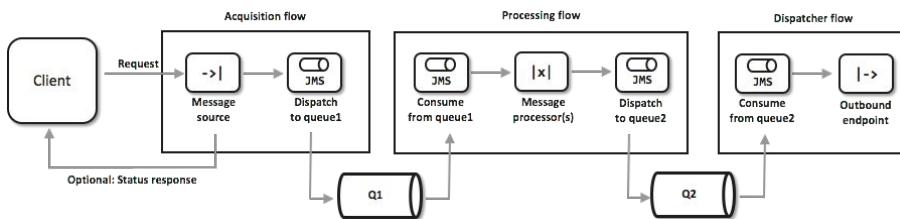
- Zero message/data loss for transactional systems is achieved using a **transaction**
- Message Ack is another way to achieve zero message loss
- Message persistence for application downtime/crash is required
- More details are provided in the persistence module



All contents © MuleSoft Inc.

16

- Zero message/data loss for non-transactional systems is achieved using a **reliability pattern**
- Splits processing between an **acquisition** flow and a **processing** flow
- The flows do not call each other directly, but use **persisted queues**
- In case of application failure, messages/data are still available in those queues



17

Understanding the two flows in this reliability pattern for non-transactional systems

- The reliability pattern consists of flows with specific responsibilities
 - The Acquisition flow**
 - Receives incoming messages then dispatches them to a persisted processing queue
 - In case of failure, processes the message until redelivery exhausted, and if that fails, then dispatches the message to a persistent error queue
 - The Processing flow**
 - Process messages from the processing queue then dispatches them to another persisted queue
 - In case of failure, processes the message until redelivery exhausted, and if that fails, then dispatches the message to a persistent error queue

Example: The acquisition flow of a reliability pattern for a non-transactional systems

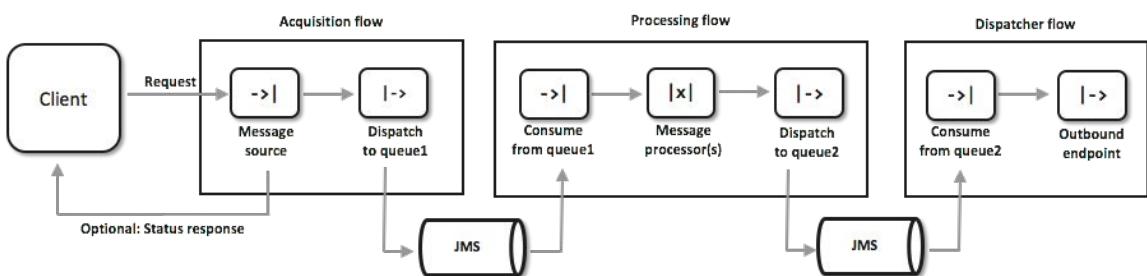


```
<!-- File listener configure with redelivery count of n for in case of failure -->
<file:listener>
    <redelivery-policy maxRedeliveryCount="n" />
</file:listener>
<!-- file is persisted in VM queue -->
<vm:publish doc:name="Publish" queueName="file"/>
<!-- In case of any error occurs, the read will be rolled back and the message processed until maxRedeliveryCount and finally persisted in error queue -->
<on-error-continue type="REDELIVERY_EXHAUSTED">
    <vm:publish queueName="fileerror"/>
</on-error-continue>
```

Achieving reliability for non-transactional systems



- The processing flow must read the message queue transactionally
- Queues can be persistent VM queues or JMS queues
- A redelivery policy is set on event sources in both flows
- REDELIVERY_EXHAUSTED type errors are handled in both the acquisition and processing flows



Other ways to achieve reliability with Mule applications



- In an earlier module you already learned about how to store state in a Mule application
 - Object Store
 - Persistent queues
 - File persistence
 - Database connector
 - Caches
 - Other external systems
- Each option has tradeoffs between reliability goals vs. performance vs. cost

Summary

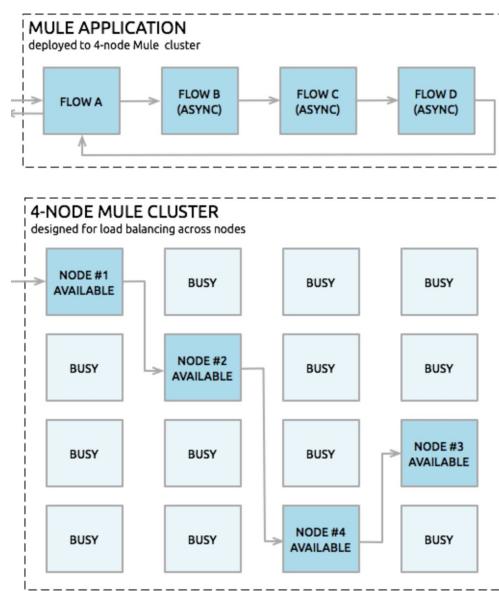


- Application level persistence (file, OSv2) achieves different levels of reliability depending on the runtime planes and configurations
- Reliability patterns for synchronous and asynchronous flows ensure reliability in the Mule application through persistent queues across various connectors

Module 13: Designing for High Availability Goals



Goal



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



- Clarify HA goals for Mule applications
- Balance HA goals with reliability and performance goals
- Identify ways to achieve high availability (HA) using Anypoint Platform, in CloudHub and on-premises
- Describe how clustering and load balancing works
- Identify HA aware connectors and their design tradeoffs

Achieving high availability (HA) goals using multiple Mule runtimes



Distinguishing between high availability (HA) vs. disaster recovery (DR) goals



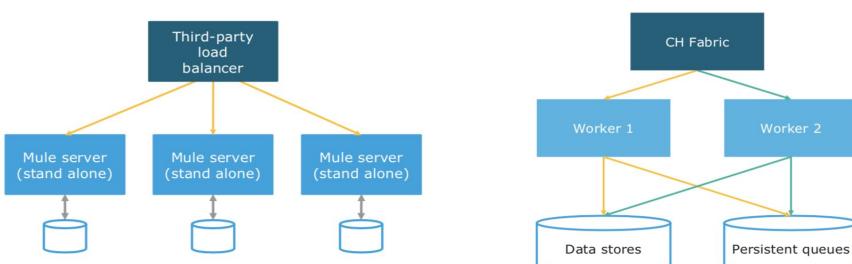
- High availability (HA)
 - How to keep the overall system operational when a system component fails
 - Usually achieved with multiple levels of fault tolerance and/or load balancing
 - In CloudHub, you can deploy a Mule application to multiple CloudHub workers
- Disaster recovery (DR)
 - How to restore a system to a previous acceptable state after a natural or man-made disaster
 - Must compensate for failure of entire sub-systems/regions/zones/data centers, and hence there are a large number of system components that failed
- Read details in the Mule User Guide

<https://docs.mulesoft.com/mule-user-guide/v/3.9/hadr-guide>

Making Mule applications highly available using Mule runtimes



- High availability can be achieved by **horizontally scaling** to multiple Mule runtimes
 - Process on multiple concurrent physical machines/VMs, often distributed across networks
- HA goals can be met via **load balancing** and/or **clustering**
 - Clustering of Mule runtimes uses an active-active model of node
 - Load distributes across the active nodes



Comparing Mule runtime server groups vs. clusters



- Runtime Manager can define server groups or clusters
- A server group is just an administrative grouping of isolated Mule runtimes
- A cluster provides additional guarantees to prevent contention between the Mule runtimes
 - File access by File based transports
 - All JMS topic subscribers connect to the same topic, resulting in duplicate processing
 - JMS request/response queues might send the response to a different Mule runtime, causing uncorrelated response processing or other failures
 - Salesforce streaming API will fail because the API only supports a single consumer

<https://docs.mulesoft.com/mule-runtime/4.1/mule-high-availability-ha-clusters>

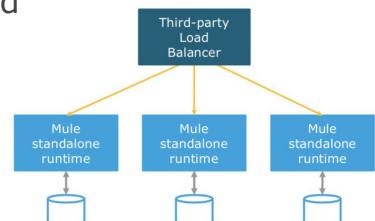
All contents © MuleSoft Inc.

7

Unclustered load balancing for HA and performance in customer-hosted runtime planes



- Multiple Mule runtimes are configured to run the same Mule application(s)
- Does not share or synchronize data between Mule runtimes
 - Could potentially lead to processing duplicate or lost messages
 - But using a shared database, message broker, or other system is possible
- Messages must be distributed or load balanced
 - Requires third party products



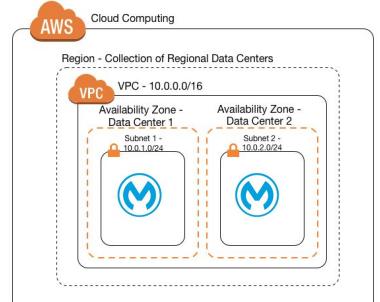
All contents © MuleSoft Inc.

8

Achieving HA in CloudHub using load balancing between multiple CloudHub workers



- HA can be achieved using multiple CloudHub workers (>1)
 - Easy to configure
 - Workers do not share any memory
 - Workers can use an external system for state management
- Each worker is created in a **different availability zone** in the **same AWS region**
- Mule application data can be stored and shared between CloudHub workers in the Anypoint Object Store (OSv2)
- VM queues can be changed to persistent or non-persistent in Runtime Manager
 - Without changing any configuration or properties of the deployed Mule app



All contents © MuleSoft Inc.

9

Achieving HA using customer-hosted clusters



- A cluster is a set of customer-hosted Mule runtimes that act as a unit
 - Servers in a cluster communicate and share information through a distributed shared memory grid
 - Selected data is replicated across memory in different physical machines
 - Cluster nodes can be configured to be more reliable so they also copy data to disk or external storage
- Nodes are aware of each other
- All cluster nodes work in active-active mode and there is a primary node where Schedulers, JMS listeners, etc. run
- To manage peak loads, nodes can be added to / subtracted from a cluster
- Only available for customer-hosted Mule runtimes

All contents © MuleSoft Inc.

10

- If one Mule runtime (node) fails, outstanding tasks transfer automatically to surviving nodes in the cluster so are still **available**
- If the failed node was the primary node than one of the remaining nodes is elected as the new primary node to continue managing HA
- Nodes can be added to a cluster to manage peak load, then later subtracted, without needing to redeploy Mule applications
- A cluster can be tuned to be either more performant or more reliable
 - The cluster can be configured with a quorum count so the cluster is not available until a minimum number of nodes are active
 - The cluster can be configured to decide how many nodes replicate the data, so data can survive if multiple nodes go offline

How customer-hosted Mule runtimes (nodes) join a cluster

- Nodes can discover and join a cluster using multicast or unicast

	Unicast	Multicast
Characteristics	<ul style="list-style-type: none">• Cluster uses IP address for identifying server	<ul style="list-style-type: none">• Cluster group servers automatically detect each other
Pros	<ul style="list-style-type: none">• No special network configuration other than IP of server	<ul style="list-style-type: none">• Nodes dynamically join the cluster when the node is started
Cons	<ul style="list-style-type: none">• IP of at least one other node must be known and configured in each node's cluster configuration	<ul style="list-style-type: none">• Only permitted in network where multicast is allowed

How shared memory is used in a cluster



- A cluster is implemented using Hazelcast to create a distributed **shared memory** data grid
 - Data is automatically replicated and available between the cluster's nodes
 - This allows data to survive if a node crashes or otherwise leaves the cluster
- Components that use a cluster's shared memory include
 - VM Queues
 - Object Stores
- Most connectors are not cluster-aware
 - But all connectors that use an Object Store are implicitly cluster-aware
 - Examples include: Cache scope, Idempotent Message validator, and the Round Robin router

Features of clustering and load balancing



- Both clustering and load balancing in active-active scenarios have pros and cons

	Clustering	Load Balancing
Pros	<ul style="list-style-type: none">• Shared, distributed memory• Ideal for HA scenarios• Built-in load balancing for VM queues• Built into Mule	<ul style="list-style-type: none">• Easy to set up• No performance overhead due to latency or data replication• Configurable load balancing algorithms (round-robin, IP sticky, load-based, etc.)
Cons	<ul style="list-style-type: none">• Performance overhead due to latency and data replication• Not supported by CloudHub• Requires 3rd-party product to achieve HTTP load balancing	<ul style="list-style-type: none">• Requires third-party product• No data synchronization• Manage idempotency programmatically

Identify cluster aware connectors and design consideration for HA



- Socket based
 - Receives incoming traffic
 - Traffic must be distributed
 - Outbound socket based connectors don't need special consideration
 - Example: HTTP
- Resource based
 - Cluster automatically manages access to resource so only one clustered instance can access resource at a time
 - Outbound (writing) resource based connector generates unique resources
 - Examples: File, FTP
 - Distributed locking is not supported while writing

Comparing storage behavior for one or more customer-hosted Mule runtimes

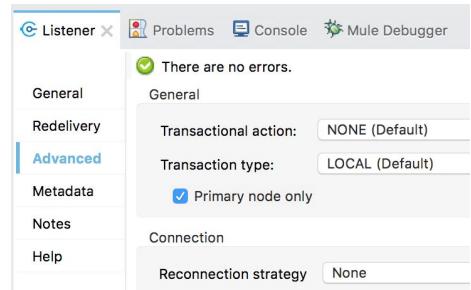


	Non-persistent Object Store	Persistent Object Store
Standalone Mule runtime	<ul style="list-style-type: none">• in-memory store in the local Mule runtime• Data does not survive server restart	<ul style="list-style-type: none">• File-based store• Data survives server restart
Customer-hosted server group	<ul style="list-style-type: none">• Isolated in-memory store per Mule runtime• Data does not survive server restart	<ul style="list-style-type: none">• Isolated file-based store per Mule runtime• Data survives server restart
Cluster of customer-hosted Mule runtimes	<ul style="list-style-type: none">• in-memory store in the local Mule runtime• Data does not survive server restart	<ul style="list-style-type: none">• Hazelcast data-grid• The cluster itself can be configured to be only in-memory, or to use file-based storage

Identify cluster aware connectors and design consideration for HA



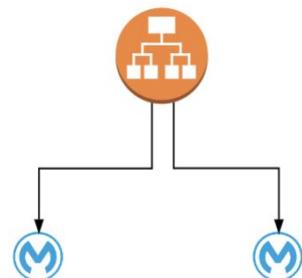
- Listener based
 - Traffic is distributed automatically
 - Must decide if the listener should only fire on the primary node or on all nodes
 - Examples: VM, JMS
- Schedulers
 - The scheduler only fires on the primary node



Load balancing for HTTP/S connector



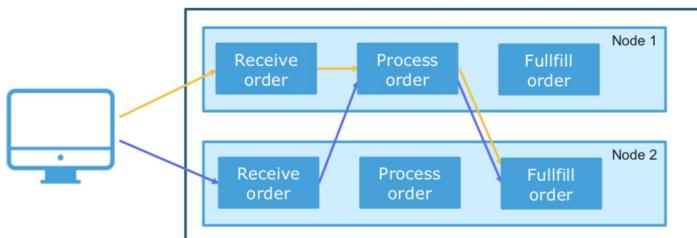
- For customer-hosted Mule runtimes, HTTP requests need to be load balanced through a 3rd-party product
 - Traffic must be distributed
 - Load balancers required (Nginx, Apache web server)
 - Outbound socket based connector don't need special consideration
- For CloudHub workers, HTTP requests are automatically load balanced through shared or dedicated CloudHub load balancers



Clustering for VM Connector



- Messages published to a VM queue in a cluster are automatically load balanced to receiving flows
 - No additional servers or infrastructure are required
 - Every node in the cluster can execute flows of deployed Mule apps
 - The cluster manager automatically determines what node to use based on load
 - Not a deterministic round-robin algorithm



All contents © Mule

19

Understanding Anypoint MQ



- **Anypoint MQ** is a multi-tenant, cloud messaging service that enables customers to perform advanced asynchronous messaging scenarios between their applications
 - Is fully integrated with Anypoint Platform
 - Offers role based access control
 - Client application management
 - Connectors
 - Optionally, can provide strict first in, first out (FIFO) processing to enable ordering of messages
 - REST API allows access by non-Mule applications
 - Displays usage statistics on the number of messages

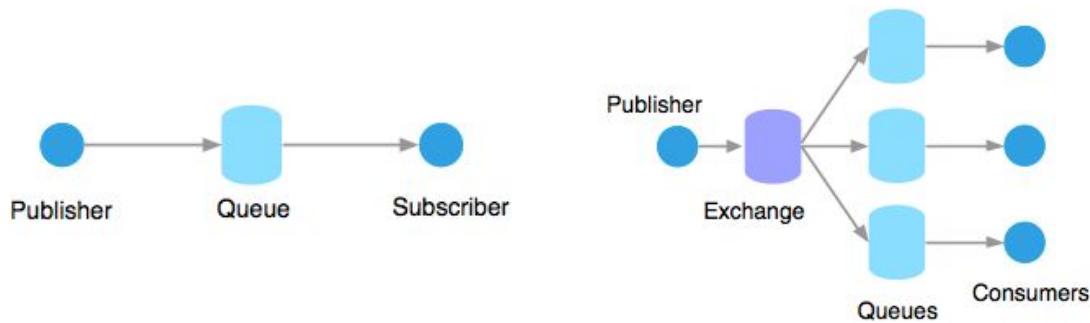
All contents © MuleSoft Inc.

20

Anypoint MQ message processing pattern



- An Anypoint MQ **queue** delivers a single message to a single consumer
- An Anypoint MQ **exchange** provides a way to broadcast a single message to many queues



How Anypoint MQ distributes messages



- In Anypoint MQ, reliability is provided via queues through a **lock and ack mode**
- A **consumer** retrieves messages from a **queue**
 - This locks the messages for a finite period of time, and makes the messages invisible to all other consumers
- If the message is not processed within the lock period, the lock times out, and consequently the message is made visible to other consumers to be processed
- This ensures that if there was a failure of some sort, such as the consumer node crashed, a message can be processed by another node

Messages in a VM queue are not load balanced for Mule apps deployed to a **standalone Mule runtime**



- Messages passed on a VM queue will NOT be automatically load balanced to receiving flows across Mule runtimes
 - Every standalone Mule runtime node will execute flow instances independently
 - The processing happens on a single node
 - No distributed processing

How VM connectors are load balanced in customer-hosted Mule runtimes



- When the Mule runtimes are combined into a cluster, persistent queues are backed by the memory grid
- Then multiple consumers share the VM queue, and messages are automatically load balanced between consumers
- The load balancing is as fast as possible, to whichever consumer is first (not round-robin)

- In **CloudHub**, each persistent VM queue is listened on by every **CloudHub worker**
 - But each message is read and processed **at least once** by only one CloudHub worker, and **duplicate processing is possible**
 - If the CloudHub worker fails, the message can be read by another worker to prevent loss of messages and this can lead to duplicate messaging
 - By default, every CloudHub worker's VM Listener receives different messages from the VM queue

Load balancing for JMS connector

- Unlike VM, JMS listener's **default behavior** is to **receive messages only in the primary node**, no matter from what kind of destination messages are being consumed
- If consuming from a queue, the **primary node** configuration can be changed to false to receive messages in all the nodes of the cluster
- Normal subscriptions where each subscriber will receive a copy of the published message, if consuming from topic with shared subscriptions mechanism (a mechanism for distributing messages to a set of subscribers to shared subscription topic), then you'll want to change the cluster configuration to consume messages **only in the primary node to false**

- A cluster automatically manages access to a resource so only one clustered node can access the resource at a time
- An outbound resource based connector generates unique resources
- Distributed locking is not supported

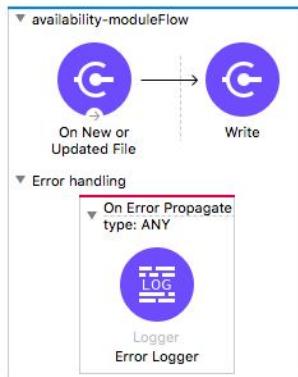
Failures of node in cluster / load balancer

- Failure of a node in a cluster or load balancer causes failures of HTTP requests processed by that node
 - However a typical enterprise level load balancer (such as AWS ELB) automatically recognizes failures of a node and stops sending subsequent requests to the failed node
- JMS, File, VM, FTP listeners, and Schedulers continue to perform processing on the elected primary node in cluster
 - The behavior is mostly consistent with the behavior in CloudHub
 - The primaryNodeOnly configuration is only used in a cluster, not by CloudHub
- Failures of a load balancer is a single point of failure
 - CloudHub load balancers are highly available and deployed in more than one availability zone

Exercise 13-1: Design an HA and reliable application for file transfer



- Design a file transfer use case that meets agreed HA goals, including minimum uptime and the minimum allowed number of Mule application failures



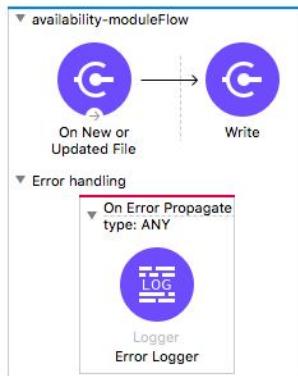
All contents © MuleSoft Inc.

30

Exercise context



- The Mule application has to transfer a file from a local directory to an online FTP server
- The Mule application has to meet an HA availability SLA of 99.99% per year and must be highly reliable against Mule application failures



All contents © MuleSoft Inc.

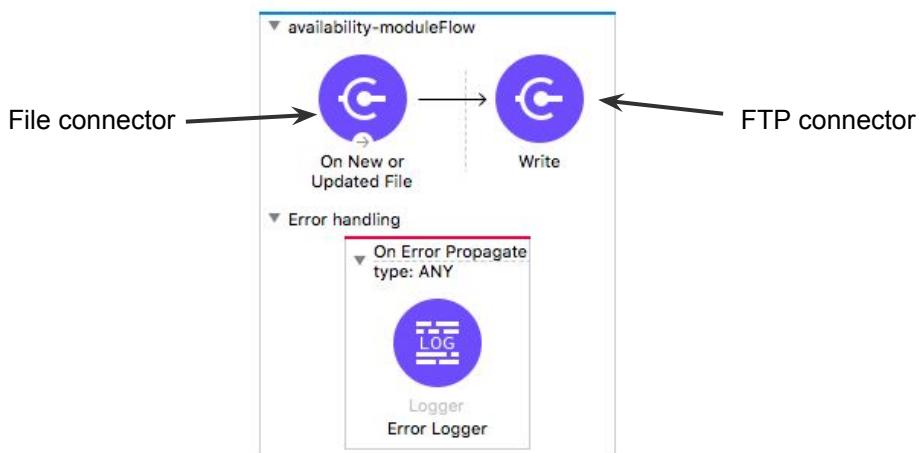
31

Exercise overview: Design an HA and reliable Mule application for file transfer



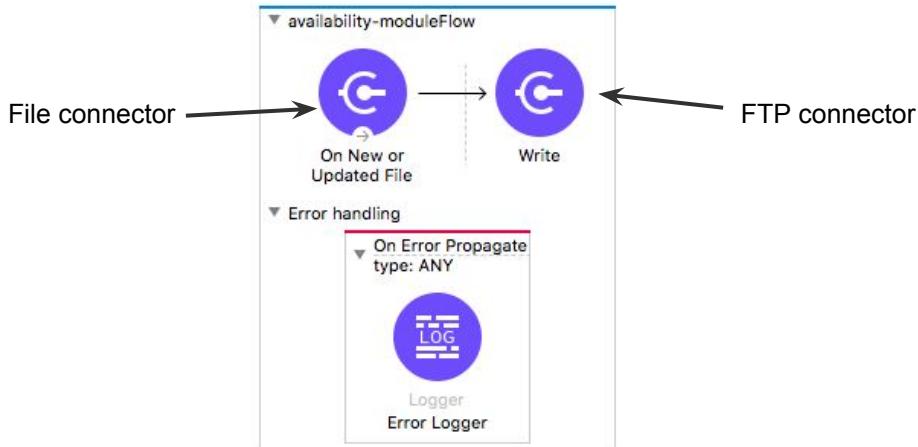
- Design flow/s for the Mule application
- Identify scaling options for the Mule application
- Identify the number of workers for the Mule application
- Decide Mule application persistence options to meet the Mule application's reliability goals
- Design the Mule application to meet reliability goals regarding Mule application and system crashes
- How would you design one flow to transfer a file to the FTP server
 - This is less reliable, but easier and more direct to implement

Exercise step: Design the file transfer flow



Exercise steps: What happens if the FTP write fails and the file is already deleted?

MuleSoft



Exercise step: Answer these questions

MuleSoft

- What failures points can reduce reliability of the system?
- If an error happens after reading the file, but before the file is completely processed, how can you assure the file is not deleted?
- What should be done if file reading fails a few times?
- What should be done if the FTP write process fails a few times?
- Can persistence help to enhance reliability?
- What are the options to maintain persistence?
- When should the file be deleted from the inbound file server?
- What horizontal or vertical scaling options can help?

Exercise solution options: Design an HA and reliable Mule application for file transfer

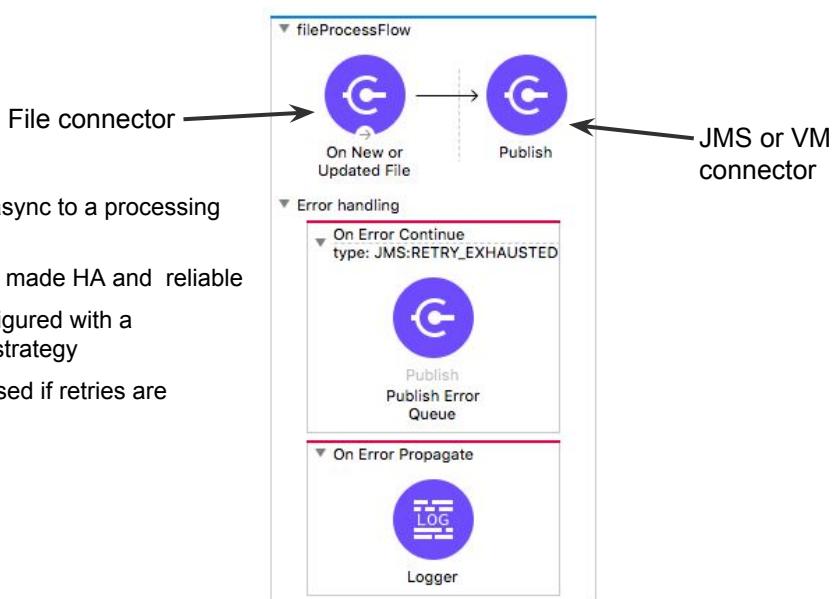


- Vertically scaling
 - A single Mule runtime does not make sense to use one node for HA (99.99%)
 - A better option is to horizontally scale the Mule application across multiple Mule runtimes
- Horizontal scaling options
 - Deploy to multiple Mule runtimes
 - Multiple nodes would access the same file
 - VM transport will not load balance between nodes
 - Clustering
 - Cluster manages resource access so only one node can access file
 - VM transport is selected for application persistence to enhance reliability in cluster

Exercise solution: How to orchestrate file transfer



- Send the file async to a processing queue
- Queue can be made HA and reliable
- Queue is configured with a reconnection strategy
- Error queue used if retries are exhausted

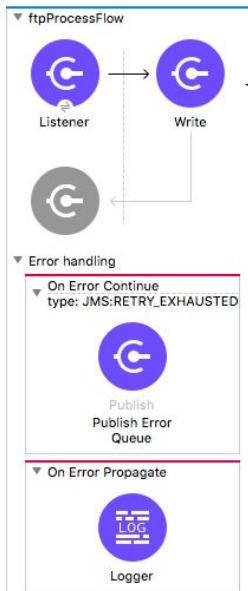


Exercise solution: How to orchestrate ftp process



JMS or VM connector

- Receive files from the processing queue
- Queue can be made HA and reliable
- Queue is configured with a reconnection strategy
- For JMS, the Listener can also configure a redelivery policy
- Error queue used if JMS redeliveries are exhausted



- Write the processed file using an FTP connector
- Set a reconnection strategy
- If the Mule application is deployed to multiple Mule runtimes (or CloudHub workers), then several files can be processed concurrently

All contents © MuleSoft Inc.

38

Summary



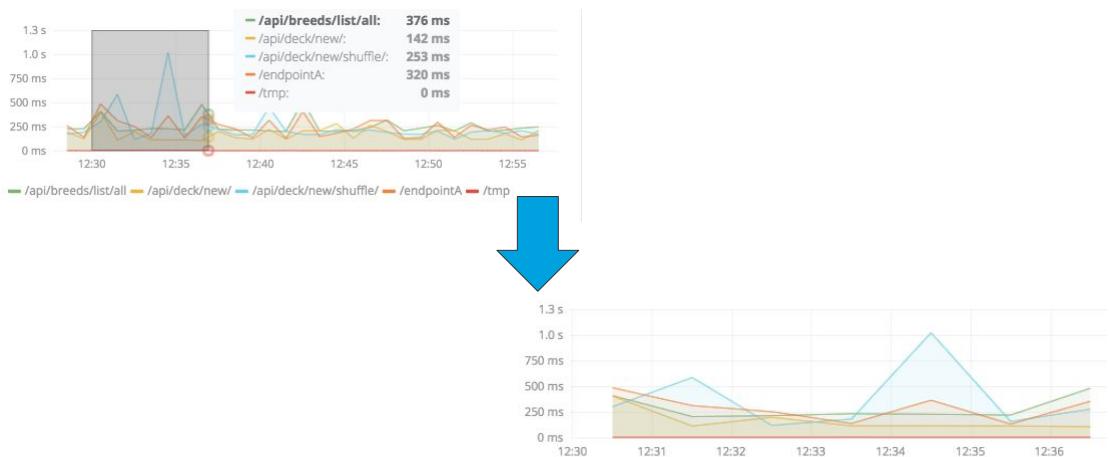
- HA goals must be clearly defined for applications
 - Performance and HA are often opposing goals, which involve trade-offs
- HA is achieved by scaling the Mule application with clustering and load balancing
- Load balancing and clustering for HTTP/S requests require an external product
- Automatic load balancing in a cluster only works for flows that start with a VM connector



Module 14: Optimizing the Performance of Deployed Mule Applications



Goal



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



- Clarify performance goals for Mule applications
- Balance performance goals with reliability and HA goals
- Identify the need for performance optimization and associated trade-offs
- Identify ways to search for and locate performance bottlenecks
- Plan, architect, design, and implement for performance
- Identify ways to measure performance
- Identify methods and best practices to performance tune Mule applications and Mule runtimes

Identify why and when to optimize Mule application performance



Why and when performance matters in a Mule application



- Some Mule applications may experience performance issues
- Performance goals are often at least partially dictated by service level agreements (SLAs)
- To optimize performance, key performance indicators must be identified and agreed upon by key stakeholders

Iterating to define and achieve performance goals



- It is important to not over-optimize performance goals
 - Performance optimization goals require additional time, money, and resources
 - Goals need to be clearly stated and agreed upon, as well as service level agreements
- It is typically recommended to tune performance in logical stages, driven by real data, tests, and tools
 - To meet, but not necessarily overly exceed performance goals

Exercise 14-1: The need for performance optimization



- Identify performance goals that may be important to particular types of Mule applications
- Identify how performance goals balance with other reliability goals
- Identify key performance indicators to measure Mule application performance objectives
- Decide which types of performance optimizations are required for a particular type of Mule application and its associated business use cases

Exercise steps



- Discuss and answer these questions:
 - Why do Mule applications need to optimize performance?
 - What are the key performance indicators for Mule application?
 - Is optimizing performance mandatory for all Mule applications?

Exercise solution



- A Mule application may experience performance issues when
 - Processing large numbers of messages
 - Processing large messages (payload size > 1MB)
 - Transforming messages extensively
 - Suffering from network latency or other performance bottlenecks

Exercise solution



- Key performance indicators for integration Mule applications
 - Throughput
 - Response time/latency
 - Capacity (number of concurrently processed messages)

Exercise solution



- Performance optimization is not always required for all Mule applications
 - Not unless experiencing performance issue
 - Use Mule runtime defaults as long as possible

"Premature optimization is the root of all evil" - Donald Knuth

How to identify performance bottlenecks



- Log analysis
 - Application log
 - Includes all log events log from the application
 - Log level govern by log setting in application or Mule runtime
 - System log
 - Includes all log events from Mule runtime
 - Log level can not govern however additional logging can enable

How to identify performance bottlenecks



- Application performance monitoring tools, including
 - The Anypoint Platform dashboards, VisualVM/JConsole, or similar
 - Commercial tools like AppDynamics, New Relic, etc.
- Monitor and manage performance and availability of Mule apps
- Detect and diagnose complex Mule application performance problems to meet SLAs
- Measure key performance indicators
- Monitoring is possible at different levels depending on each tool's capability

Profiling Mule applications to identify performance bottlenecks



- Application profiling in Mule applications is often performed for two reasons
 - Memory issues
 - To detect memory leaks or excessive load situations
 - A Java heap memory dump can help to analyze these types of issues
 - Application unresponsiveness
 - To detect blocked threads, long-running threads, and waiting threads related to the Mule application
 - Thread dumps can help analyze the issue

- Before starting performance tuning, verify the Mule application already functions as expected
- Guarantee a stable performance testing environment
 - Guarantee all external systems work within predefined SLAs
 - Or mock expected data and simulate external input SLAs, and external response SLAs accordingly
 - Guarantee the testing tools and environment do not introduce additional load that will skew or bias collected data and observations
- Select appropriate tools for performance measurements
- Make sure **performance KPIs** are **clearly defined** and **agreed** by stakeholders

Example: KPIs to be measured to performance tuning Mule applications

- Average and worst case payload size
- Processing latency and CPU load at each Mule component
- Time spent between each network hop
- Time spent between Mule components
- Throughput
 - Requests or transactions per unit of time, e.g., 100 requests/second
- Latency or response time
 - Unit of time at a given load, e.g., 100 milliseconds mean +/- 20 milliseconds standard deviation at 100 requests/second

Example: Tools to measure performance



- Advanced Rest Client, SoapUI, JMETER to generate load
- Profiling tool such as YourKit or VisualVM to identify symptoms
- Other larger load testers to test performance at expected production scales
- Anypoint Monitoring
- MUnit to test individual components or parts of a flow

Example: Monitoring traffic and KPIs



- Common tools include
 - Anypoint Monitoring, Anypoint Visualizer, and other Anypoint dashboards
 - AppDynamics
 - New Relic
 - Nagios
 - Spunk or ELK
 - Zipkin or Jaeger

Architecting for Performance



Architecting for performance

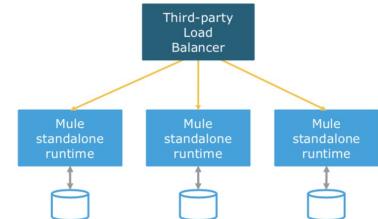


- Particular performance goals may be achieved using Mule runtime and Mule application **scaling options**
 - Vertical scaling
 - Scale Up (for example, increase the vCore size on CloudHub)
 - Provide more resources (CPU cores, RAM) on each machine
 - Main use case: performance
 - Horizontal scaling
 - Scale out
 - Process on multiple machines
 - Improves both performance and availability
 - Load balancing and/or clustering need to be decided

HTTP request load balancing for performance



- Multiple Mule runtimes execute the same Mule application(s)
- When not clustered, there is no data sharing/synchronization between Mule runtimes and Mule applications
 - Could potentially lead to duplicate message processing or lost messages
 - Using a shared datastore is possible
- Messages must be distributed/load balanced
 - Requires third party products (on-prem only)
 - Deployed apps on multiple workers are load balanced in CloudHub



Clustering for performance



- A performance profile can be configured inside the Mule application or Mule runtime cluster for high performance
 - Disabled by default
- Configure a Mule runtime cluster for performance by setting the storeprofile value in the mule-cluster.properties or wrapper.conf file
`mule.cluster.storeprofile=performance`
- Or override per Mule application in a configuration global element

```
<mule>
    <configuration>
        <cluster:cluster-config>
            <cluster:performance-store-profile/>
        </cluster:cluster-config>
    </configuration>
</mule>
```

- Setting the performance profile has two effects
 - Disables distributed VM queues, using local VM queues instead to prevent data serialization/deserialization and distribution in the shared data grid
 - Implements node local in-memory object stores instead of shared memory grid architecture (Hazelcast) to avoid replication
- It is a common misconception that applications always perform better in a cluster!

Reviewing features of clustering and load balancing

- Both clustering and load balancing in active-active scenarios have pros and cons

	Clustering	Load Balancing
Pros	<ul style="list-style-type: none">• Shared, distributed memory• Ideal for HA scenarios• Built-in load balancing for VM queues• Built into Mule	<ul style="list-style-type: none">• Easy to set up• No performance overhead due to latency or data replication• Configurable load balancing algorithms (round-robin, IP sticky, load-based, etc.)
Cons	<ul style="list-style-type: none">• Performance overhead due to latency and data replication• Not supported by CloudHub• Requires 3rd-party product to achieve HTTP load balancing	<ul style="list-style-type: none">• Requires third-party product• No data synchronization• Manage idempotency programmatically

- HTTP vs HTTPS
 - The latency difference between HTTP and HTTPS requests are only about 0.2 to 0.4ms more than HTTP requests (Mule 3.* stats), after the initial TLS handshake
- TLS
 - Use latest version of TLS 1.2.
 - Older version of TLS has poorer performance and vulnerabilities
- VM vs JMS
 - VM in memory protocol performs better than JMS
 - Transient queue performs better than persistent in VM and JMS

Sizing CloudHub workers to meet performance goals



How Mule applications are deployed to multiple CloudHub workers



- CloudHub automatically distributes multiple workers for the same Mule application across two or more availability zones for maximum reliability
- When a Mule application is deployed to two or more workers
 - The HTTP load balancing service automatically distributes requests across all the CloudHub workers in an approximately **round-robin** manner
- A Mule application can be scaled out to a maximum of 8 workers or 16 vCores

Configuring autoscaling in CloudHub



Autoscaling in CloudHub



- Runtime Manager provides an **autoscaling** feature
 - Only certain enterprise licenses support autoscaling
 - Must ask MuleSoft support to enable this feature
- Allows Mule runtimes to **scale** in response to CPU or Memory usage thresholds being exceeded
- Each autoscaling policy can decide to either
 - Increase or decrease the current Mule runtime vCore size (vertical scaling)
 - Increase or decrease the current number of Mule runtimes (horizontal scaling)

General

Name: myPolicy

Scale based on: CPU Usage

Rule

Scale up: If CPU Usage is above 80 % for more than 10 minutes.

No other scaling policy will be applied for 30 minutes.

Scale down: If CPU Usage is below 20 % for more than 10 minutes.

No other scaling policy will be applied for 30 minutes.

Action

Modify: Number of workers

Limit between: 1 and 4 workers

Restrictions of the Anypoint Platform autoscaling feature



- Each autoscaling policy is only triggered every 30 minutes
- Trigger conditions are limited to memory or CPU usage

- Write your own scripts to deploy or undeploy Mule runtimes or CloudHub workers based on various triggering conditions
 - Perhaps using the Mule Maven plugin
 - Combine with other Anypoint Platform REST APIs
- Triggering conditions could be
 - Alerts (CPU, memory, ...)
 - Monitoring results (Anypoint Monitoring)
 - Behavior of requests/transactions submitted specifically for autoscaling purposes

Designing for Performance



- Different Mule components have different performance behaviors
 - Synchronously
 - When a client produces a request, the client waits for a response from the consumer
 - Usually has better performance for moderate throughput and small payloads
 - Examples: HTTP Listener, Web Service Consumer, JMS Publish consume operation
 - Asynchronously
 - Request and response are separate interactions, and the response is optional
 - Producer does not wait for response from consumer
 - High throughput and large payload
 - Examples: JMS Publish operation, Async scope

Performance features of common Mule components

- Batch
 - Processes messages in batches
 - Useful for streaming input or synchronizing/processing collections of data in batches of records at a time
 - High Throughput
- Scheduler
 - Scheduler runs periodically for new data
 - Useful for batch processing
 - Use a short scheduling interval to keep the source and target in sync
- Event/messaging queues
 - Events/messages decouple data producers from data consumers
 - Useful for real time data integration

- Streaming
 - Data can be read multiple times or accessed randomly from stream using the DataWeave expression language
 - Data can be sent to multiple places without the need to cache that data in memory first.
 - Can transparently process larger-than-memory data
 - SaaS providers often have restrictions on accepting streaming input. Rather, use streaming batch processing when writing to a file such as CSV, JSON, or XML.
 - Slows the pace at which it processes transactions

Exercise 14-2: Identify the best processing model for optimizing performance

- Identify ways to sync data between an enterprise database system and Salesforce
- Make optimal design decisions for three different data synchronization scenarios
 - Daily periodic data synchronization
 - High volume data synchronization
 - Real time lower latency data synchronization of smaller volumes of data

Exercise scenarios: Identify the best processing model for optimizing performance



- The application has to sync data between a MySQL source database and a target Salesforce system
- Design a flow for 3 different scenarios and explain merit for your decision
- Three different scenarios
 - Scenario 1 - Sync 500 records a day with periodic sync
 - Scenario 2 - Sync more than 10,000 records within 5 minutes
 - Scenario 3 - Real time sync between MySQL and salesforce (less than 10 records a minute)

Exercise step: Design flows to meet the Scenario 1 requirements



- Scenario 1: Sync 500 records a day with periodic synchronization
 - Identify expected and required workload for the scenario
 - Does the scenario need real time processing or can periodic processing suffice?
 - Identify how much latency is allowed before new database records must be synchronized to Salesforce
 - Identify message source(s) for flow(s) that best meet workload and processing requirement (real time, periodic, etc.)
 - Evaluate various processing options for scenarios
 - Async message queues, batch, schedulers, etc.
 - Evaluate and analyze if the expected and required workloads justify the use of the proposed processing model

Exercise solution: Design flows to synchronize 500 records a day with periodic synchronization

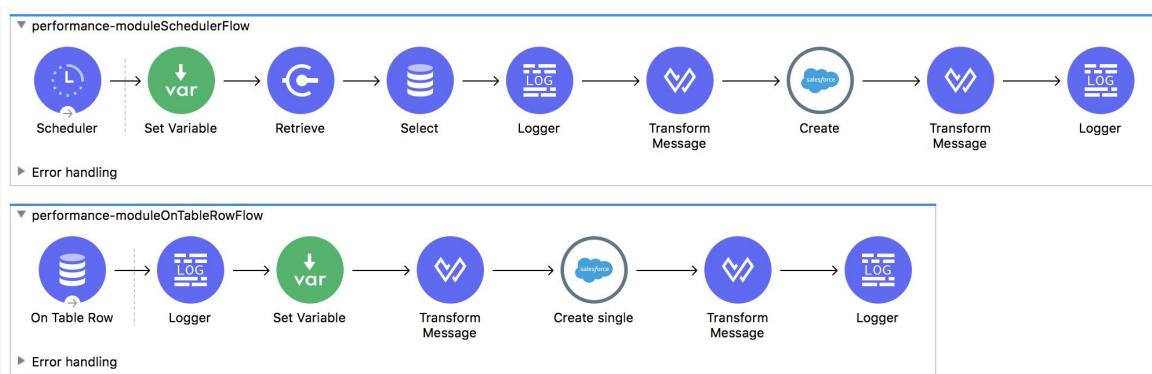


- This is low throughput data sync scenario and data sync is not required in real time
- An appropriate processing model is
 - Scheduled polling of the source database
 - Works for periodic sync of data
 - Low to high throughput of data
 - Smaller number of records should not require streaming data
 - All 500 records can be transformed and processed at once in the Mule app's memory
 - Event triggered by source database changes
 - Allows real time data sync with moderate throughput of data

Exercise solution: Sync 500 records a day with periodic sync



- Periodically select from a database table, then transform and sync (write) each retrieved row/record to Salesforce in a bulk operation, and process the results
- Also poll the same database table, then transform and sync (write) each retrieved row/record in a single Salesforce Create operation



Exercise step: Design flows to meet the Scenario 2 requirements



- Scenario 2 - Sync more than 10,000 records within 5 minutes
 - Identify expected and required workload for the scenario
 - Does the scenario need real time processing or can periodic processing suffice?
 - Identify how much latency is allowed before new database records must be synchronized to Salesforce
 - Identify message source(s) for flow(s) that best meet workload and processing requirement (real time, periodic, etc.)
 - Evaluate various processing options for scenarios
 - Async message queues, batch, schedulers, etc.
 - Evaluate and analyze if the expected and required workloads justify the use of the proposed processing model

Exercise solution: Sync more than 10,000 records within 5 minutes

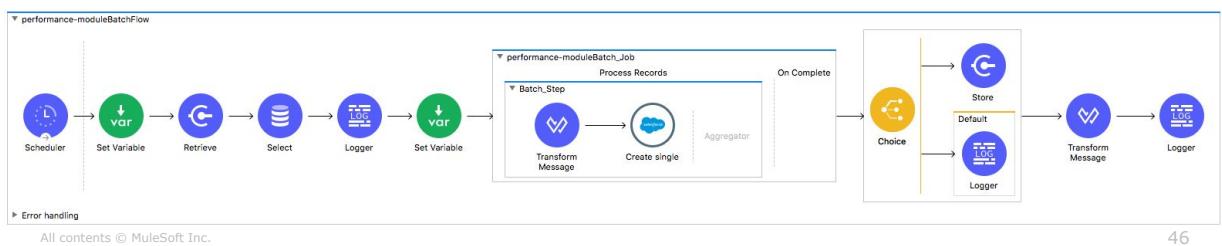


- This is a high throughput data synchronization scenario and data synchronization is required in batches
- The likely processing model is to combine
 - Scheduler
 - Works for periodic high frequency sync (every 30 secs)
 - Can handle high data throughput (can handle large streaming data)
 - Batch
 - Can handle high data throughput (can handle large streaming data)
 - Records can be processed in parallel in multiple threads
 - Useful when data synchronization is required to complete as fast as possible

Scenario 2 - Sync more than 10,000 records within 5 minutes



- Retrieve the last processed record id from an object store
- Periodically select from a database table
- Transform and sync (write) batches of database records in a batch job
- The object store is used to only create new records in Salesforce, to avoid duplicate work
 - New Salesforce record IDs are then stored in the object store



46

Exercise step: Design flows to meet the Scenario 3 requirements



- Real time sync between a source database and Salesforce
 - Identify expected and required workload for the scenario
 - Does the scenario need real time processing or can periodic processing suffice?
 - Identify how much latency is allowed before new database records must be synchronized to Salesforce
 - Identify message source(s) for flow(s) that best meet workload and processing requirement (real time, periodic, etc.)
 - Evaluate various processing options for scenarios
 - Async message queues, batch, schedulers, etc.
 - Evaluate and analyze if the expected and required workloads justify the use of the proposed processing model

Exercise solution: Design flows for real time sync between a source database and Salesforce

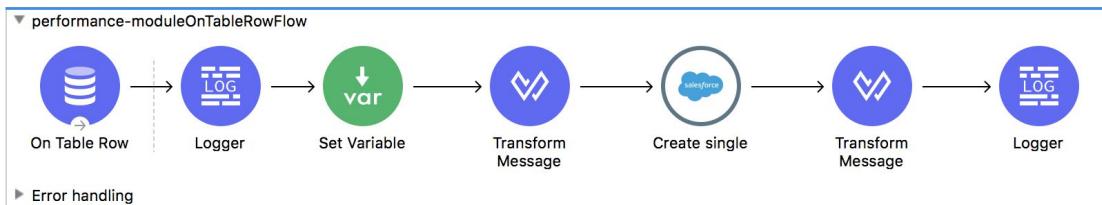


- This is low throughput data sync scenario and data sync is required in real time
- An appropriate processing model is
 - Let the database trigger new events
 - Low throughput of data
 - When data sync is required in real time

Scenario 3 - Real time sync between salesforce and MySQL



- Use the On Table Row event source to poll for new rows in the source database table
 - This is not true real time, but the polling interval can be set to a low value, and a watermark and avoid duplicate processing of existing rows
 - Only new rows are sent over the network to this Mule flow
 - Individual rows are processed one at a time then created in the target Salesforce system



Implementing Mule applications to meet specific performance goals



Implementing for performance



- Transformation
 - Avoid unnecessary conversions to Java data types, and work with data directly
 - Let the Mule handle streaming for you
 - Do not convert binary data types to String, Byte[] or InputStream unless you are performing a direct integration with custom Java code.
 - DataWeave uses default memory buffer of 1572864 bytes, if transformation need excess memory then it uses system's hard disk as a buffer
 - DataWeave buffer size can changed by setting system property com.mulesoft.dw.buffersize and assign it the number (in bytes)

- Scatter-Gather
 - Parallel execution of routes greatly increases the efficiency of your application
 - Max Concurrency sets concurrency for parallel execution of scope

- Logging
 - Defaults to async for performance
 - Can log errors synchronously and the info,debug level asynchronously
 - GC log can be captured

```
<!-- log4j2.xml in MULE_HOME/conf for Mule classes -->  
    <AsyncLogger name="org.mule" level="INFO"/>  
    <AsyncLogger name="com.mulesoft" level="INFO"/>  
<!-- wrapper.conf in MULE_HOME/conf -->  
    wrapper.java.additional.<n>=-Xloggc:%MULE_HOME%/logs/gc.log
```

- Network latency
 - Large payload
 - Compress large payload (gzip)
 - Can use the Compression module
 - Caching
 - Cache responses

Measuring Mule application performance



How to measure Mule application performance



- First identify related KPIs
- Identify tools that can effectively and efficiently measure these KPIs
- Measure CPU and memory utilization of a Mule application
- Decide if performance optimization is required and the associated trade-offs
- Profile the Mule application then tune the Mule application to optimize resource utilization
- Continue measuring to validate performance goals are met

Exercise 14-3: Measure Mule application performance



- Tune a Mule flow and a batch job to optimize performance
- Identify tools that can effectively and efficiently measure agreed upon KPIs
- Profile and tune a Mule application to optimize resource utilization
- Validate performance goals are met

Exercise overview



- Measure the key performance indicators identified in exercise 14-1
 - Identify tools useful for measuring KPI for application
 - Decide whether performance improvement is required for application
 - Identify CPU and memory utilization for demo app
 - Optimize resource utilization

Exercise steps: Measure performance

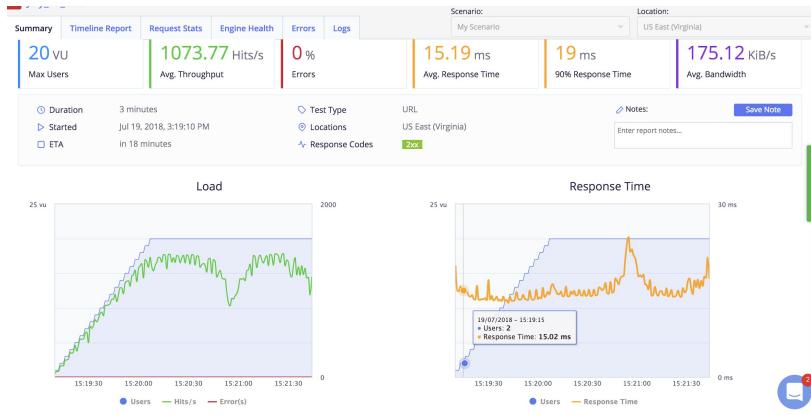


- Have you used any tools in past to measure the performance of system?
- Check whether a tool is able to measure the KPIs for the application
- Analyse KPI against SLA requirement
- Do you need performance improvement in application
- Does any monitoring tool on Anypoint platform will help to measure CPU and memory utilization for application
- Can we optimize resource utilization based on CPU and memory optimization
- Can we form a rule for resource optimization

Exercise solution: Measure performance



- Identify tools useful for measuring KPI for application
 - JMeter or BlazeMeter or any open source performance testing tool



All contents © MuleSoft Inc.

61

Exercise solution: Decide whether performance improvement is required



- Compare your KPI against your SLA
- If SLA is not met, then performance improvement is required

All contents © MuleSoft Inc.

62

Exercise solution: Measure performance



- Identify CPU and memory utilization for demo app
 - Anypoint Monitoring provides built-in dashboards
 - Overview, Inbound, Outbound, Performance, Failure, JVM and Infrastructure
 - Overview Dashboard provides below details for application -
 - Total Inbound Requests
 - Average Response Time Inbound
 - Total Outbound Requests
 - Average Response Time Inbound
 - CPU Utilization
 - Memory Utilization
 - Thread Count - Server
- Anypoint Monitoring docs
 - <https://docs.mulesoft.com/monitoring/dashboards>

Exercise solution: Measure performance



- Optimize resource utilization
 - Identify CPU and memory utilization on 0.1 vCore(smallest vCore for CloudHub)
 - Deploy more than one instance of application for high availability
 - If CPU and memory utilization is above 70% then scale horizontally/vertically
 - Optimize vCores for application based on resource utilization
 - Auto scaling of CloudHub workers is option(only for special licensed customers)

Performance tuning Mule Applications



Performance tuning Mule applications



- Auto-tuning of thread pools in Mule runtime

Pool	Min	When?	Max	Increment
CPU_LITE	#cores	Mule startup	2 * #cores	1
CPU_INTENSIVE	#cores	Mule startup	2 * #cores	1
BLOCKING_IO	#cores	Mule startup	#cores + ((mem - 245760)/5129)	1
GRIZZLY (shared)	#cores	Deployment of first app using HTTP Listener	#cores + 1	1
GRIZZLY (dedicated)	#cores	Deployment of first app using HTTP Requester	#cores + 1	1

Performance tuning features and considerations using HTTP/S connectors



- HTTP connector
 - Uses HTTP persistent connections by default
 - Set keep-alive header true with HTTP 1.0 client
- HTTPS connector
 - Use latest version of TLS 1.2.
 - Older version of TLS has poorer performance and vulnerabilities

Performance tuning features and considerations using JMS connectors



- JMS Connector
 - Caching is on by default
 - Caches JMS sessions/consumers and producers
 - Use sessionCacheSize to adjust size of JMS cached session
 - Disable JMS message persistent at JMS server
 - Configure numberOfConsumers on JMS listener for high throughput or else sequential message processing
 - Configure ACK mode to meet your SLA (discuss in detailed transaction module)
 - Avoid durable subscriber and message filtering

Performance tuning features and considerations using JMS connectors in clusters



- Set the `primaryNodeOnly` to false for queue listeners so consumers on all cluster nodes can receive message **from** the **queue**
- Set `primaryNodeOnly` to false for topic listeners so all consumers can receive message from **topic with shared subscription**
- A shared subscription to a JMS topic ensures that the different replicas of the Mule application on different cluster nodes receive different JMS messages from the topic

Performance tuning features and considerations using a database connector



- Database connector
 - Caching is on by default
 - Caches db connections
 - Use bulk operation based DB connector for batch processing
 - Streaming
 - Enable to start processing large result sets
 - Max rows and fetch size
 - Example: max rows = 1000, fetch size = 200 **limits** the response to max 1000 rows, and the database will try to stream at most 200 rows at a time to the database connector (so in this case would require 5 separate network round trips)

Performance tuning features and considerations using a VM connector



- VM connector
 - Use for HA
 - For performance, prefer flow references within the same Mule app instead of VM endpoints

Performance tuning features and considerations using Batch jobs



- Batch jobs
 - Default block size is 100
 - Max Concurrency sets concurrency for batch jobs (default is twice the available cores in CPU)
 - Run comparative batch sizes to find optimum concurrency for the use case. If you are processing 200 records with default block size of 100, to process 2 records in parallel, you need concurrency of 2.
 - Use Fixed Size Batch Aggregator to do bulk operation for supported connectors(db, salesforce)

Performance tuning features and considerations using a streaming Batch Aggregator scope

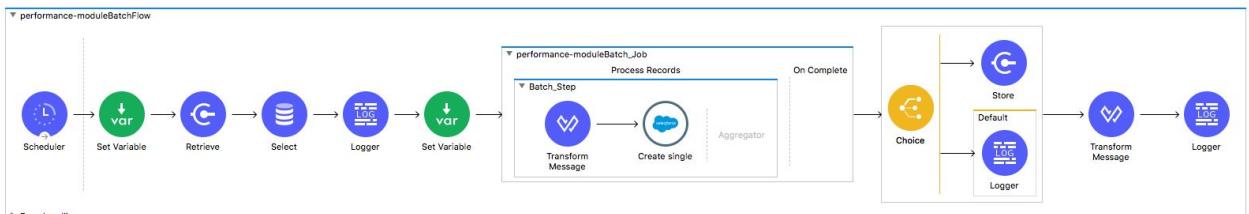


- Streaming Batch Aggregator
 - Receives all the records in the job instance without running out of memory
 - Random access of processing records is **not supported** for streaming aggregators
 - SaaS providers often have **restrictions** on accepting streaming input
 - Rather, use streaming batch processing when writing to a file such as CSV, JSON, or XML
 - Slows the pace at which it processes transactions

Exercise 14-4: Performance tune a Mule application



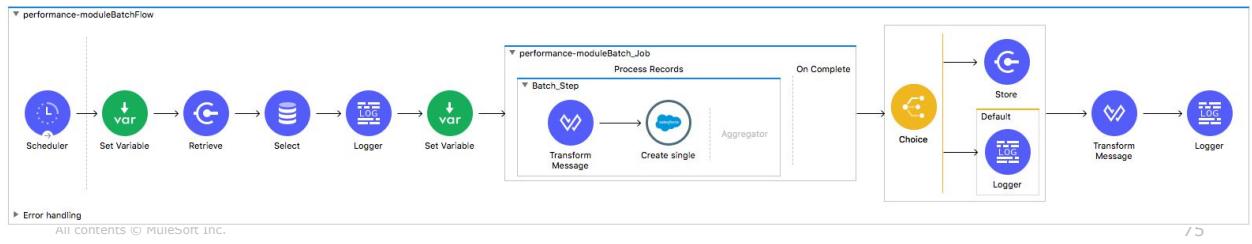
- Tune a Mule flow and a batch job to optimize performance



Exercise steps



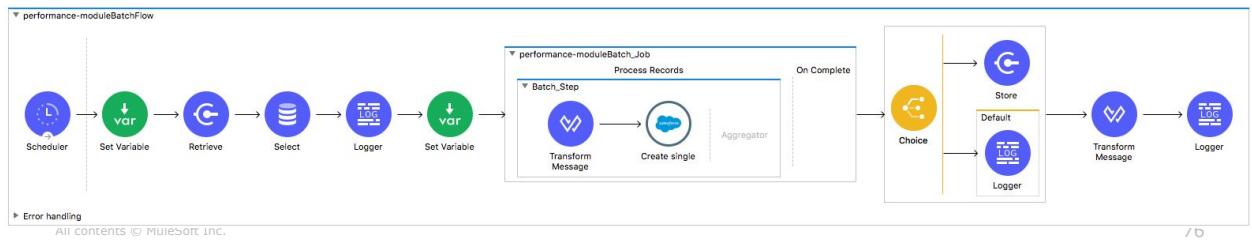
- In Scenario 2 of Exercise 14-2, you designed a flow to sync data between an enterprise database system MySQL and Salesforce
- Import project from student files exercise-14-4.jar



Exercise steps



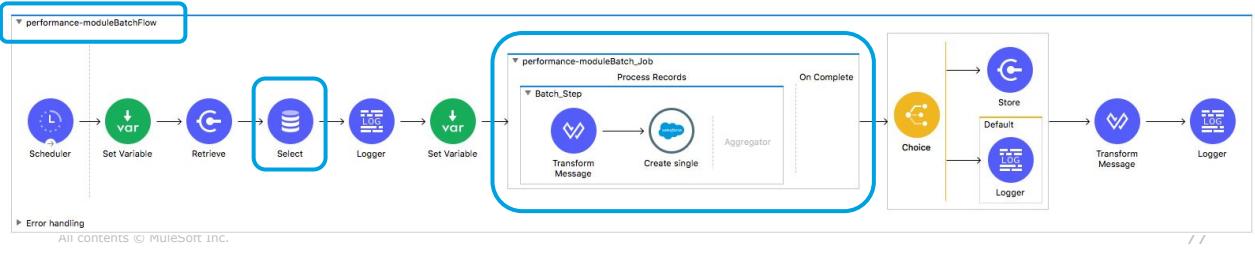
- Tune the flow and its components for optimum performance
 - Select Mule components that may benefit from performance tuning
 - Identify tuning parameters for the selected Mule components
 - Analyze the trade-offs of each tuning parameter within the context of the entire Mule application



Exercise solution: Select Mule components that may benefit from performance tuning



- Flow
- DB Connector
- Batch Job
- Batch Aggregator



Exercise solution: Identify tuning parameters for the selected Mule components

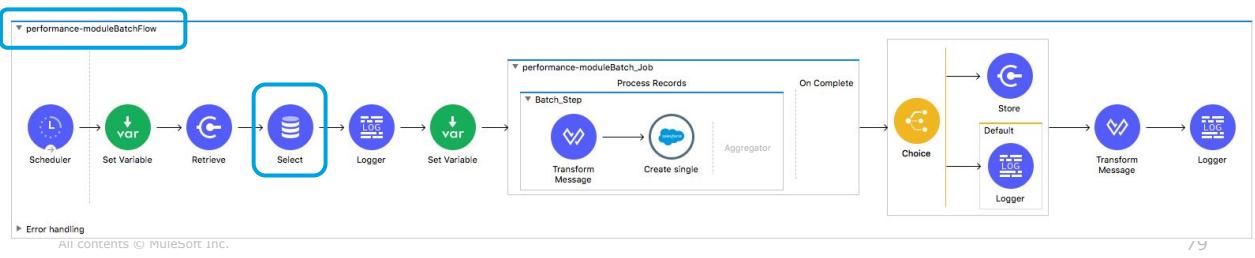


- Flow or components
 - max concurrency
 - The max number of simultaneous invocations that component can receive at any given time
- DB Connector
 - streaming, max rows, fetch size
 - Limit total results from a DB SELECT, then fetch/stream them in smaller batches
- Batch Job
 - max concurrency, block size
- Batch Aggregator
 - streaming, fixed size batch

Exercise solution: Analyze tuning parameter trade-offs for flows and connectors



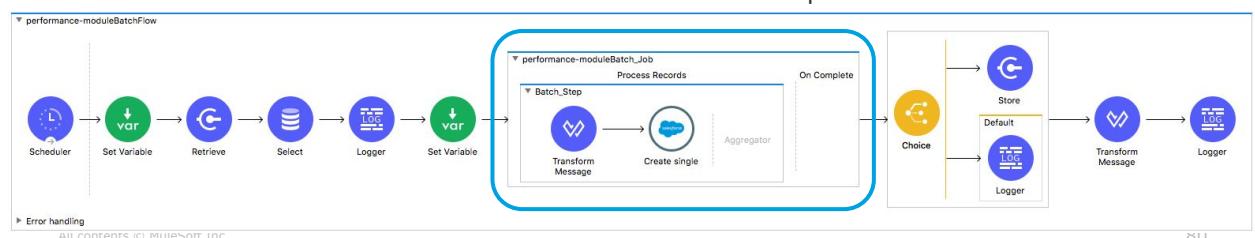
- Analyze the trade-offs of each tuning parameter within the context of the entire Mule application
 - Flow
 - Flow concurrency is not required for the batch job use case
 - DB Connector
 - Streaming is option, but not required, for this use case
 - Max rows with fetch size can create incremental batches



Exercise solution: Analyze tuning parameter trade-offs for batch jobs



- Analyze the trade-offs of each tuning parameter within the context of the entire Mule application
 - Batch Job
 - Run comparative batch block sizes to find optimum concurrency for the use case
 - Batch Aggregator
 - Fixed size batch aggregator suits for use case
 - Streaming with SaaS provider has restriction
 - Salesforce API has limitation of max 250 records per bulk insert

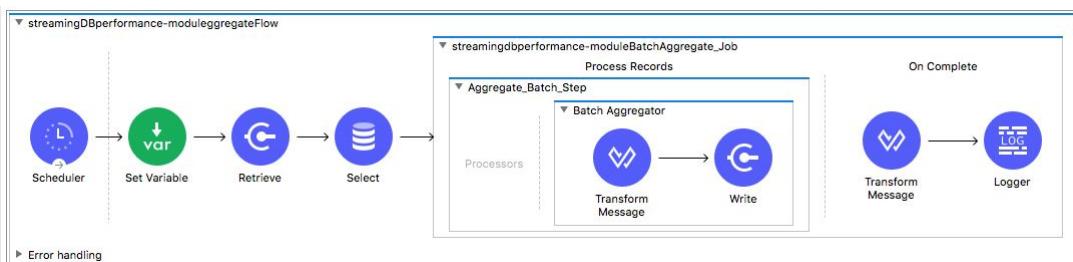
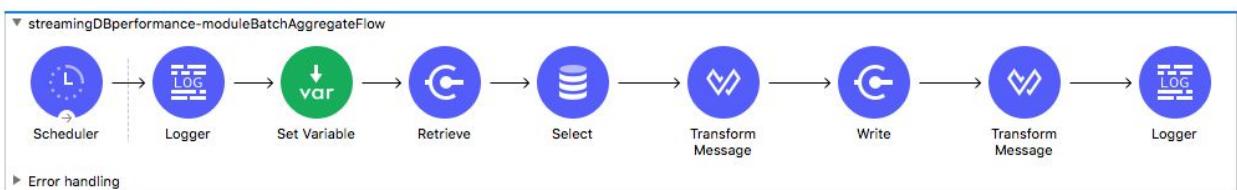


Exercise step: Performance tune a high volume Mule application



- Design flow for write 1M records from enterprise database system MySQL into JSON file. Tune the flow/batch for optimum performance
 - Identify component to performance tuning
 - Identify tuning parameter for component
 - Understand trade-off of each tuning parameter

Exercise solution: Performance Tuning of Mule application



Exercise solution: Performance Tuning of Mule application



- Identify Mule application components that should be tuned for performance
 - Flow
 - DB Connector
 - Batch Job
 - Batch Aggregator
 - File

Exercise solution: Performance Tuning of Mule application



- Identify tuning parameter for the Mule application's components
 - Flow
 - max concurrency
 - DB Connector
 - streaming, max rows, fetch size
 - Batch Job
 - max concurrency, block size
 - Batch Aggregator
 - streaming, fixed size batch
 - File
 - Lock

Exercise solution: Performance Tuning of Mule application

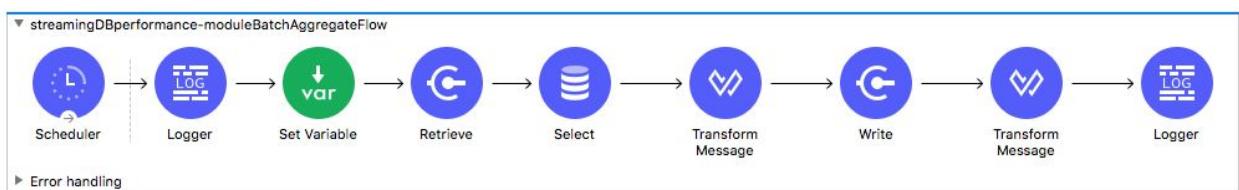


- Understand the trade-offs of each tuning parameter
 - Flow
 - Flow concurrency is not required otherwise file write need synchronization
 - DB Connector
 - Streaming is required for use case
 - Max rows with fetch size can create stream, fetch size is mandatory for streaming
 - Batch Job
 - Run comparative batch block sizes to find optimum concurrency for the use case
 - Batch Aggregator
 - Streaming batch aggregator suits for use case as writing to file
 - File
 - File lock is disabled by default however writing to file using multiple thread may create deadlock on file.

Exercise solution: Performance Tuning of Mule application



- Flow with DB streaming works best for the use case
 - Flow
 - Flow concurrency is not required otherwise file write need synchronization
 - DB Connector
 - Streaming is required for use case
 - Max rows with fetch size can create stream, fetch size is mandatory for streaming
 - Max rows = 1M and Fetch Size = 500 work best



- Batch Aggregator streaming slows pace of transaction compare to non streaming aggregator however reduces memory footprints
- It advisable to have steaming on for Batch Aggregator if prior processing using streaming however it is not mandatory
- Batch processing improves performance with non repeatable streaming

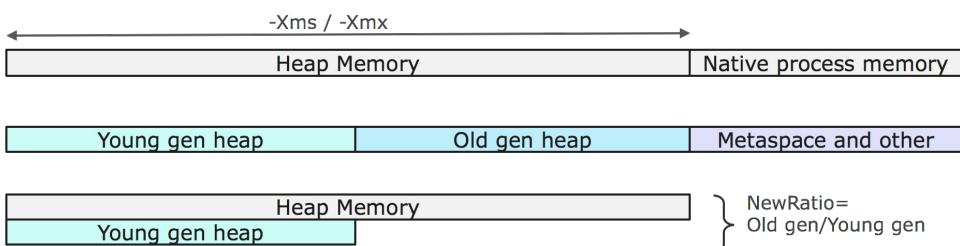
Performance tuning a Mule Runtime



Performance tuning for customer-hosted Mule runtimes



- Mule 4 uses JDK 1.8
 - Java Objects reside in **heap memory**
 - New objects are created in **young generation heap memory**
 - **Garbage collectors** move objects to **old generation heap memory**
 - **Metadata memory** resides natively outside the JVM heap memory
 - Used for class metadata such as class descriptors, methods, bytecodes, classes and classloaders
 - By default, uses all available memory on the host



All contents © MuleSoft Inc.

89

Performance tuning for Mule(customer hosted)



- The JVM that starts a Mule runtime can be tuned with standard JVM system properties
 - Can be set in the Mule runtime's wrapper.conf file, located in \$MULE_HOME/conf
 - CloudHub uses HotSpot, the standard Oracle JVM
 - Set the initial and maximum heap size to the same value (default 1024 MB)
 - Set the **NewRatio** to set the ratio of the old and young generation heaps (default 1)

```
wrapper.java.additional.N=-XX:MetaspaceSize=256m
```
 - Set the **MaxMetaspaceSize** and **MetaspaceSize** to limit the amount of native memory used for class metadata (default 256 MB)

```
wrapper.java.additional.N=-XX:MetaspaceSize=256m
```

```
wrapper.java.additional.N=-XX:MaxMetaspaceSize=256m
```

All contents © MuleSoft Inc.

90

- The JVM garbage collector (GC) is selected and tuned using standard JVM system properties
 - Can be set in the Mule runtime's wrapper.conf in \$MULE_HOME/conf
- Parallel GC
 - Default GC for Oracle HotSpot JVM
 - Enough memory and large number of CPU
 - Optimized for throughput
- Concurrent-Mark-Sweep (CMS) GC
 - Uses more memory and CPU
 - Response time for application is crucial

Thread pools used by Mule runtimes

- Global thread pools are used to execute all flows for all Mule applications
- Three thread pools are used to execute individual event processors, based on the event processors execution type
 - CPU-intensive, CPU-light, or IO-intensive
- Some event processors internally flag their execution type
- Otherwise the Mule runtime automatically selects an execution type based on all the other event processors in the flow

- Threads and concurrency parameters can be tuned in the scheduler-pools.conf located in \$MULE_HOME/conf
 - Only accessible in customer-hosted Mule runtimes
 - The **cpuLight** and **cpuIntensive** thread pool size is twice the number of vCores
 - The **IO** thread pool core size is the same as the number of cores
 - The **IO** thread pool max size is governed by the number of cores and maxMemory
 - $\text{cores} + ((\text{mem} - 245760) / 5120)$ for 4 vCores worker with 1024 MB max memory ~ 152
- Except for rare edge cases, MuleSoft makes a strong recommendation to **not touch this file** and **stay with the default values**

Performance tuning individual flows and components

Component	Tuning Parameter
Flow	Max Concurrency
Batch Job	Max Concurrency, Batch Block Size
Batch Aggregator	Streaming, Aggregator Size(both are mutually exclusive)
Scatter-Gather	Max Concurrency

- Reconnection strategy applies to all connectors

Connector	Tuning Parameter
HTTP Connector	Use persistent connections
JMS Connector	Session cache size, cache consumer, cache producers, ack mode, max redelivery
JMS Listener	No of consumers, ack mode, max redelivery
DB Connector	Max pool size, Min pool size, acquire Increment, prepare statement cache size, transaction Isolation
DB Operation	Streaming, timeout, max rows, fetch size (required for streaming)

Summary



- Performance optimization can be performed at the architecture, design, development, runtime, and OS/VM levels
- Load balancing and clustering may help to improve performance, but not always
- The JVM instance can be tuned for Mule runtimes installed in customer-hosted infrastructure
- JVM tuning options include heap and metaspace sizes, memory tuning parameter, garbage collection choices
- Streaming adds latency to flow executions, but also reduces memory footprints
- Strong Recommendation : Go with DEFAULTS

Mule runtime tuning guidelines and resources

- Java tuning guides
 - <https://docs.oracle.com/javase/8/docs/technotes/guides/vm/qctuning/>
 - https://docs.oracle.com/cd/E21764_01/web.1111/e13814/jvm_tuning.htm#PERFM151
 - <https://www.slideshare.net/mulesoft/mule-runtime-performance-tuning>



Module 15: Designing Secure Mule Applications and Deployments



Goal



Secure Properties Config

General Advanced Notes

Name: Secure_Properties_Config

General

File: config-http.yaml

Key: \${encryption.key}

encrypt

Algorithm: AES (Default)

Mode: CBC (Default)

TLS Context

Global TLS Configuration

General Notes

Generic

Name: TLS_Two_Way_Server_Context

Trust Store Configuration

Type: JKS

Path: clientStore.jks

Password: XXXXXXXX Show password

Algorithm:

Insecure: False (Default)

Key Store Configuration

Type: JKS

Path: serverStore.jks

Alias: selfsigned

Key Password: XXXXXXXX Show password

Password: XXXXXXXX Show password

Algorithm:

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



- Identify Anypoint Platform security concepts and options
- Describe how to secure APIs on Anypoint Platform
- Understand the security needs addressed by Anypoint Platform Edge security
- Differentiate between MuleSoft and customer responsibilities related to Anypoint Platform security
- Evaluate security risks for Mule applications
- Describe how to secure Mule application properties
- Describe how to secure Mule application data in transit

Introducing Anypoint Platform security concepts and options



- Securing administrative access to various Anypoint Platform features
- Securing access to deployed Mule applications
 - Securing Mule application connectors
 - Securing the networks to which Mule applications are deployed
- Securing Mule application property placeholder values
- Securing APIs with policies
- Securing data at rest
- Securing data in motion across network edges

Controlling access to Anypoint Platform features



- **Roles** can be defined to limit or allow access by various Anypoint Platform users to various Anypoint Platform features
 - Anypoint Platform users can be assigned to roles, and then those users get all the access permissions from those combined roles
 - This allows for distributed administrative groups
 - There are some predefined roles, and other custom roles can also be created
- Every Anypoint Platform username is assigned to exactly one Anypoint Platform **organization**
 - A particular user can be invited to switch to a different organization

An organization owner is the super user of the Anypoint Platform organization

- The **Organization Owner** is the first username to sign up for the organization in Anypoint Platform
 - This is not a role, but a super user identifier for this one user
- The organization owner inherits the **Organization Administrator** role by default
 - The Organization Administrator role has every possible permission
 - Other users can be added to the Organization Administrators role
- Each organization has a **client id** and **client secret** to secure communications with the Anypoint Platform REST APIs

How an identity provider (idP) can be tied in with an Anypoint Platform organization



- By default, Anypoint Platform performs its own identity management
- One external idP can instead be integrated with the Anypoint Platform organization
 - Roles and access control are still enforced inside Anypoint Platform

Managing business groups, users, roles and environments



- Organization has business groups, child business groups under business groups, roles, environments and users
- Pre-configured role that business group owners acquire automatically
- Manages business groups
 - Business groups has its client id and client secret
 - Provides isolation of resources
 - But the same user can also be assigned to other roles in different business groups
 - vCores are assigned at business groups makes those vCores available only to the business group and unavailable to the parent organization
 - Business groups has its own environments
 - Deleting business group is NOT recoverable as all resources get deleted

Managing access to business groups



- Access to resources can be controlled by granting appropriate roles in that business group
- To obtain membership to a business group, a user needs to be invited and granted a role
 - When adding users to a role, any user in the organization is eligible to be added
 - Custom roles can be created in a business group

Cloudhub Admin [Sandbox]

Description: [Cloudhub \(Sandbox\) Admin users](#)

Permissions Users

Name	Username
Q Add a user by name or email...	
Cloud01 Instructor	Cloud01Instructor
Cloud01 Student20	Cloud01Student20 OWNER

Using an external identity management server with Anypoint Platform



- Can secure Anypoint Platform control plane by configuring
 - OpenID Connect: End-User identity verification by an IdP including SSO
 - SAML 2.0: Web-based authorization including cross-domain SSO
- OpenID Connect supports
 - PingFederate
 - OpenAM
 - Okta
- SAML supports
 - PingFederate
 - OpenAM
 - Okta
 - And many others

Client management options



- Can apply an OAuth 2.0 policy to authenticate client applications
- Anypoint platform supports one IdP for client management per organization
 - PingFederate
 - OpenAM
 - OpenID Connect Dynamic Client Registration (OIDC DCR) compliant identity providers
- Client access for an API is granted for a specific environment and API

This organization is not currently using external identity management.
Read [documentation](#) for additional information or start configuring:

Identity Management ▾

Client Management ^

OpenAM

OpenID Connect Dynamic Client Registration

PingFederate

All contents © MuleSoft Inc.

13

Responsibility for Mule runtime security



	MuleSoft-hosted control plane	Customer-hosted control plane
MuleSoft-hosted runtime plane (CloudHub iPaaS)	<ul style="list-style-type: none">• MuleSoft secures both planes• Customer secures Mule apps• Customer is responsible for securing communication between CH and on-prem systems	<ul style="list-style-type: none">• N/A
Customer-hosted runtime plane	<ul style="list-style-type: none">• MuleSoft secures control plane• Customer secures the runtime plane and Mule apps• Shared responsibility for securing communication between on-prem Mule runtimes and control plane• Can configure customer-hosted Mule runtimes to be FIPS 140-2-compliant (does not apply to CloudHub)	<ul style="list-style-type: none">• Customer secures both planes• Customer is responsible for securing communication between Mule runtimes and on-prem systems• Can configure the Mule runtime to be FIPS 140-2-compliant

All contents © MuleSoft Inc.

14

- Anypoint Runtime Fabric
 - Is a variant of the customer-hosted runtime plane managed by the MuleSoft-hosted control plane
 - MuleSoft is responsible for securing the Anypoint Platform control plane
 - Customer secures Mule runtime and applications
 - Shared responsibility for securing communication between Runtime Fabric and control plane

Securing Mule application endpoints



How secure communications are typically supported over the internet



- **Secure communication** is the safe sharing of information that cannot be intercepted by a third party "in the middle"
- Supported using
 - Cryptography
 - Symmetric cryptography
 - Client and server share the same key to encrypt and decrypt
 - Asymmetric cryptography
 - Server issues a public key to the client allowing it to encrypt the message
 - Server keeps a private key which is the only key that can decrypt the message; one key to lock the message and another key to unlock it
 - Digital certificates
 - Uses public/private key certificates signed by trusted authority or self signed for communications

Secure communication protocols supported by MuleSoft

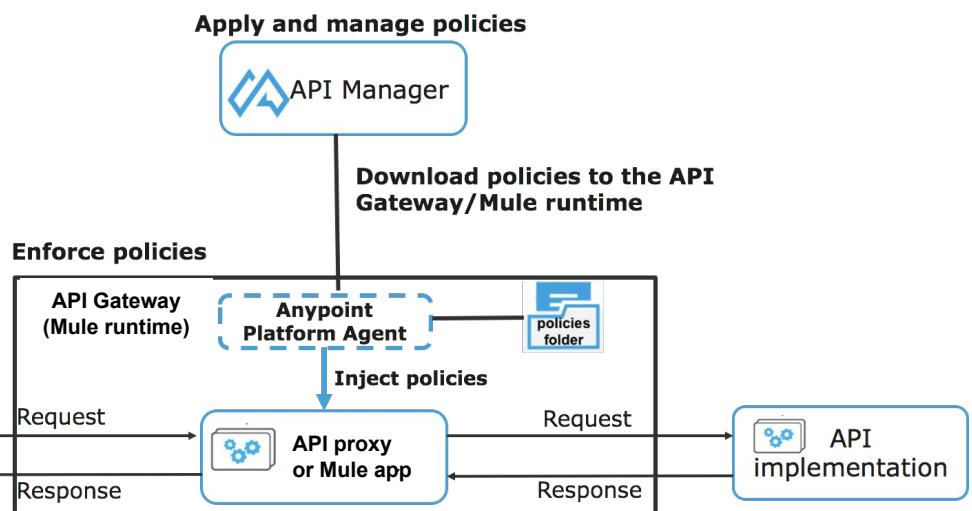


- Mule applications and Anypoint Platform support secure communication using
 - Symmetric and asymmetric cryptography standards
 - Digital certificates
- Supported protocols include
 - HTTPS, TLS, SFTP, FTPS, SMTP/S, IPSec

Securing APIs with policies using API Manager



How policies are enforced using API Manager



API policies apply non-functional requirements to an API



Policy Type	Usage
Client ID enforcement	<ul style="list-style-type: none">Requires authorized client applications to use a client id and client secret
CORS control	<ul style="list-style-type: none">Interacts with API clients for Cross-Origin Resource SharingRejects HTTP requests whose Origin request header does not match configured
Authentication/Authorization	<ul style="list-style-type: none">OAuth 2.0 token enforcement API policiesBasic Authentication: LDAP/SimpleIP-based access control<ul style="list-style-type: none">Blacklisting, whitelisting
Payload threat protection	<ul style="list-style-type: none">Guard against attacks sending over-sized HTTP request bodies
Quality of Service (QoS)	<ul style="list-style-type: none">Rate Limiting: Rejects requests above limitSpike Control: Queues requests above limit
Caching, logging, and others	<ul style="list-style-type: none">Log parts of the message before or after an API proxy or corresponding backend service is invokedCaching policies cache the entire backend HTTP response in the API proxyHeader injection and removal policies can be applied to the inbound request or outbound response messages

All contents © MuleSoft Inc.

21

How policies are enforced using API Manager



- When a policy is configured in the API Manager, the policy is downloaded to the **connected API Gateway or Mule runtimes, inside a /policies folder**
 - Some policies require modifying the RAML API definition. In such cases the API proxy has to be **redeployed** to the API Gateway or Mule runtime
- The policies are then injected into the API proxy or Mule app
 - This way policies can be dynamically changed without re-deploying the API proxy or Mule applications (when the RAML API definition does not change)
 - Does not require restarting the API Gateway or Mule runtimes
- API autodiscovery must be enabled within the application with the API ID
- API client needs permission to access resource protected through API policies

All contents © MuleSoft Inc.

22

Automated API policies for all MuleSoft-hosted Mule runtimes



- Per API policy, API Manager can configure automatically applying a policy to every CloudHub worker of particular version(s)

The screenshot shows the API Manager interface with a modal dialog titled 'Select Exchange Policy'. The dialog has two dropdown menus: 'All Categories' and 'All Mule Versions'. Below these is a list of 'Policies' with their descriptions. The policies listed are:

- > Client ID enforcement
- > Cross-Origin resource sharing
- > Detokenization
- > OAuth 2.0 access token enforcement using Mule OAuth provider
- > Header Injection
- > Header Removal
- > Basic authentication - Simple
- > HTTP Caching
- > IP blacklist

At the bottom right of the dialog are 'Cancel' and 'Configure Policy' buttons.

All contents © MuleSoft Inc.

23

Exercise 13-1: View an API's policies



- View an API's policies using API Manager
- View possible policy types and categories available in Anypoint Exchange

All contents © MuleSoft Inc.

24

Exercise steps



- View an API's policies in API Manager
 - Login to anypoint.mulesoft.com and go to API Manager
 - View American Flights API version v1
 - Look at the existing policies applied to the API

Exercise steps



- View possible policy types and categories available in Anypoint Exchange
 - View all the types of policies available in API Manager from Anypoint Exchange which can be applied to API
 - Identify the types of API categories available in Anypoint Exchange
 - Identify the types of policies available for each API category

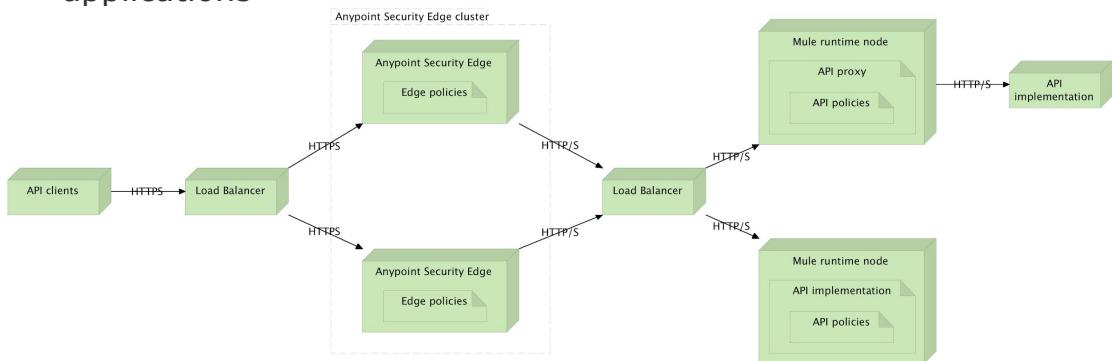
Identifying Anypoint Security options



Types of API policies available through Anypoint Platform



- You have already seen how API policies can be defined in API Manager and then injected into Mule applications
 - At runtime, these policies are enforced directly inside a Mule application
- **Anypoint Security** offers other types of policies that can be enforced at the edges of your network for multiple Mule applications



How edge security is used by Mule applications

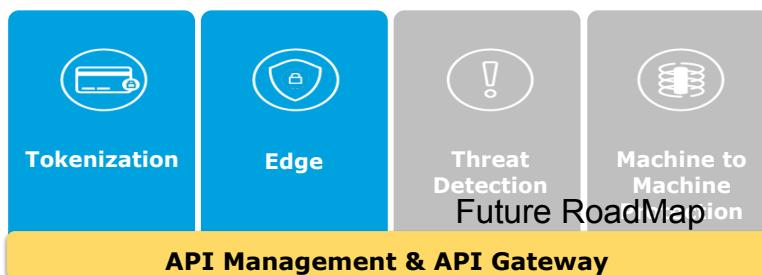


- Anypoint Security Edge Security is an enterprise solution for **perimeter (edge) level security of APIs**
- Anypoint Security is a **standalone product** deployed outside Mule runtimes
- It is typically deployed in a **DMZ** in a customer-hosted runtime plane
- These edge security policies are applied
 - To inbound requests after they are sent from a calling API client, but before they arrive at the Mule application API endpoints, so **before** API Manager defined policies are applied
 - To outbound responses back to the calling API client, after leaving the Mule application API endpoint, so **after** API Manager defined policies are applied

Anypoint Security



- Adds additional security policies to protect APIs on the edge of your network
- Uses external security servers and services to enforce advanced security related policies
- These API policies are independent of the API policies defined in Anypoint Platform API Manager



Comparing which Anypoint Security options are supported by various runtime planes



	Customer-hosted	CloudHub	Anypoint Runtime Fabric
Edge Security	Yes	No	in 2018
Tokenization and masking	Yes	No	in 2018
Edge Encryption/Decryption	Yes	No	No

Edge policies vs API policies



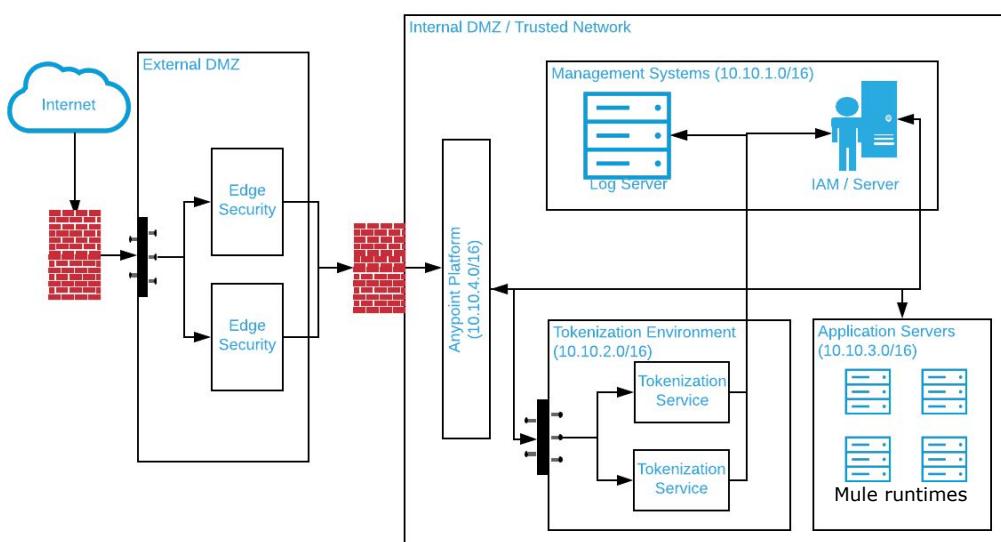
	Edge policies	API policies
Where policies are applied	Applied in the Edge gateway server	Applied in the API implementation or API proxy app
How the policy implementations works	A single Edge policy can apply to many API instances	A single API policy can apply to exactly one API instance (deployment)
Available policies	<ul style="list-style-type: none">• Service Virtualization• Connection Security and Certificate Management• Content Security• Quality of Service• Application Level(Dos)	<ul style="list-style-type: none">• Client ID enforcement• CORS control• Authentication/Authorization• Payload threat protection• Quality of Service (QoS) <p>https://anypoint.mulesoft.com/exchange/?type=policy</p>

Where Anypoint Security servers are usually deployed

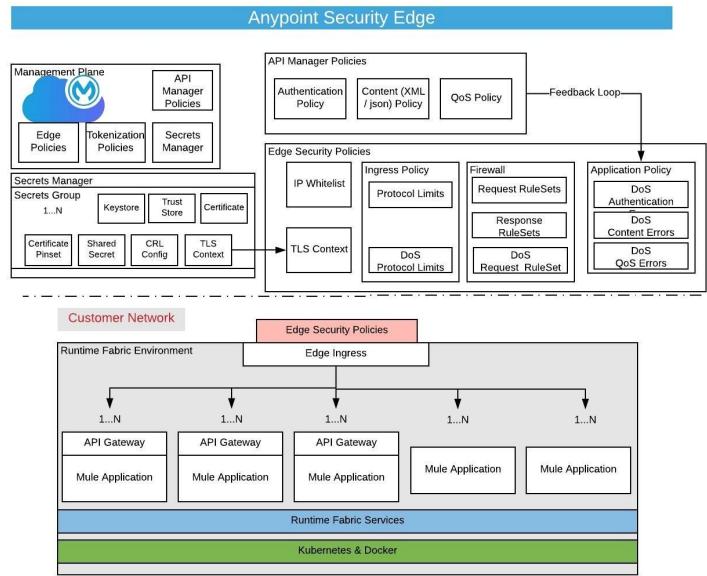


- **Edge Security servers** are usually deployed in a **DMZ** in a customer-hosted environment
 - Provides **edge policies**
- **Tokenization** is usually deployed **inside the firewall** to replace sensitive data with fake data in the same format
 - Such as credit card numbers, social security numbers, or phone numbers

Understanding the interactions between Anypoint Security servers



Anypoint Security Edge



All contents © MuleSoft Inc.

35

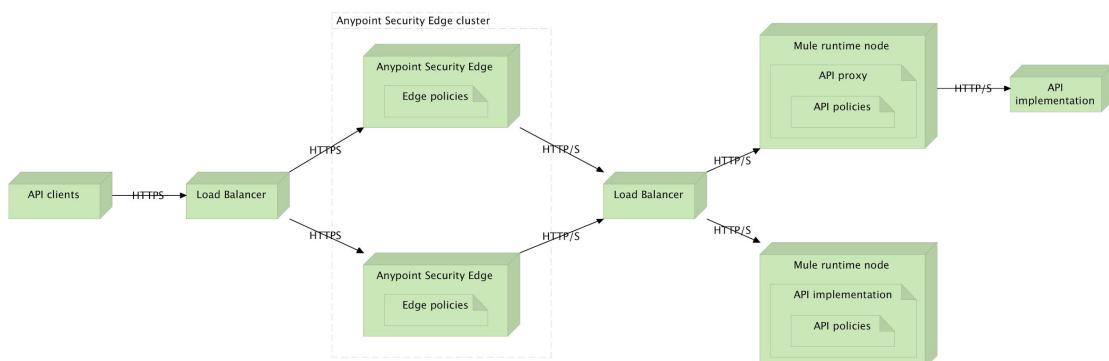
Identifying Anypoint Security edge security features and options



- A policy is a mechanism for **enforcing filters** on traffic
- These filters generally control things like
 - Compliance
 - Security
 - Quality of service (rate-limiting, throttling)
 - DoS
- Define custom policies by creating **policy definition** and **policy configuration files**
- Apply multiple policies and set the order of their execution

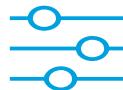
Edge security policies can be changed dynamically at runtime

- Just like API policies defined in API Manager
 - The policies can be dynamically changed without redeploying the API applications
 - Does not require restarting the Mule runtimes



Anypoint security - Edge security overview

 MuleSoft



Service Virtualization
<ul style="list-style-type: none">Edge Proxy for REST and SOAP

Connection Security and Certificate Management
<ul style="list-style-type: none">SSL / TLS TerminationCertificate ManagementChain of Trust Validation / ConfigurationCRL Configuration

Content Security
<ul style="list-style-type: none">Message inspection for Malicious Content, Injection AttacksHeader ValidationMessage Size LimitsMessage Attachment Size limits

Quality of Service
<ul style="list-style-type: none">Resource Consumption Management / Quota & Throttling

Application Level (DoS)
<ul style="list-style-type: none">Network Level Denial of Service Protection against too slow or too overwhelming requestsRate Limiting / Shaping of Traffic (Peak Protection)Request/Response inspection

All contents © MuleSoft Inc.

39

Anypoint security - Edge security features

 MuleSoft

- Edge Proxy Design policies
 - Abstracts away internal service with a secured endpoint (host/port)
 - Content Security (CAP)
 - Quality of Service (QoS)
 - Denial of Service (DoS)
 - Global Input Server - Support
 - Message Mediation Policy
 - Routing Policy
- Connection & Transport Security
 - SSL/TLS Termination
 - Server Name Indication (SNI)
 - Certificate Pinning
 - Symmetric / Asymmetric Keys configuration
 - Certificate Management
 - Chain of Trust Validation / Configuration
 - CRL configuration
 - Certificate Lifecycle

All contents © MuleSoft Inc.

40

Anypoint security - Edge security **content attack prevention (CAP)**

 MuleSoft

- Prevents malicious content from reaching a service during runtime execution
- Inspects messages for
 - Pattern Scans
 - SQL injections, XPath injections, and DTDs
 - Limit Message Size
 - Limit # of file attachments / size of each attachment
- Policy will generate a CAP violation if the request matches any of the above criteria

Anypoint security - Edge security **quality of service (QoS) policy**

 MuleSoft

- Manages resource consumption for a user-defined resource, such as a SOAP web service, REST API, or authentication service
- Some sample system resources that are tracked include
 - Request data rate
 - Raw request data rate
 - Response data rate
 - Failed Response rate
 - Message buffer utilization
- Each QoS policy creates a single QoS tracking record for the identity, such as AAA identity, IP address, etc.

Anypoint security - Edge security **denial of service (DoS)** policy

MuleSoft

- All policies mentioned earlier provide information to the DoS policy and can be configured centrally for the following actions
 - **Alert** - Sends a message to administrators that a DoS event occurred
 - **Block-Interval** - Rejects messages from a client within an administrator-defined for a period of time
 - **Block-forever** - Permanently block messages from a client unless the administrator removes the restriction
 - **Shape-Interval** - Restricts the rate at which it will accept messages from a client based on a administrator-defined rate.
 - **Shape-forever** - Restricts the rate at which it will accept messages from a client based on a administrator-defined rate unless the administrator removes the restriction

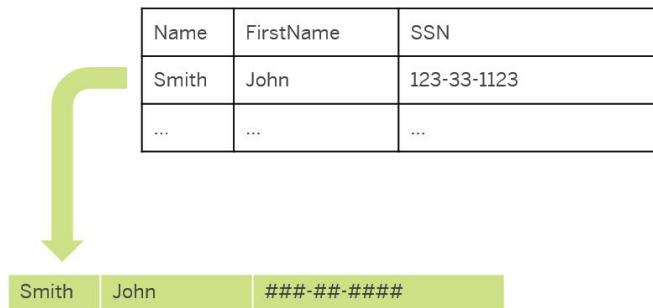
Identifying Anypoint Security encryption and masking features and options



How masking works



- **Masking** protects your data by transforming it into a readable format that's useless to anyone who steals it
- The actual data is replaced by placeholder information
- There is no algorithm to revert the data to its original state
- The real data was replaced and is gone forever

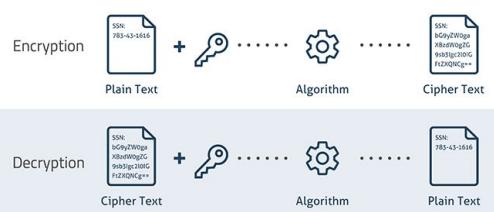


Encrypting data with a symmetric key



- In **symmetric key encryption**, one key is used to both encrypt and decrypt the information
- There can be security related issues
 - If the key is compromised, it can be used to unlock, or decrypt, all of the data it was used to secure
 - How can you securely exchange a symmetric key over a public network?

SAMPLE ENCRYPTION AND DECRYPTION PROCESS

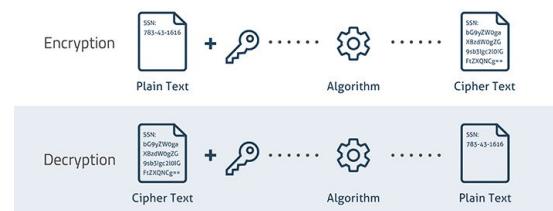


Using asymmetric key (also called public-key) encryption



- **Asymmetric key encryption** was developed to allow multiple parties to exchange encrypted data without needing to manage or exchange the same encryption key
 - The public key can be freely distributed since it is only used to lock the data
 - The private key is used to decrypt the data

SAMPLE ENCRYPTION AND DECRYPTION PROCESS



All contents © MuleSoft Inc.

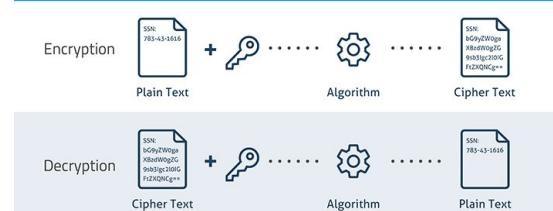
47

In practice, asymmetric encryption is usually only used to safely exchange symmetric keys



- Asymmetric encryption is much slower and more CPU intensive than symmetric key encryption
 - Asymmetric encryption is often used to safely exchange a **symmetric key between two parties** (such as a client and a server)
 - The symmetric key is then used to encrypt and decrypt message payloads
 - A new symmetric key may be periodically exchanged for added security

SAMPLE ENCRYPTION AND DECRYPTION PROCESS



All contents © MuleSoft Inc.

48

Identifying Anypoint Security tokenization features and options



Other Anypoint security features

MuleSoft // MeetUp

Tokenization

Application data validation logic work "as is"
No downstream application changes needed
Compliance scope reduction (PCI, HIPAA, GDPR)



Encryption

Information anonymization
Analytics without exposing sensitive data



Data Masking

Sensitive data obfuscation
One way process: Cannot get original value back



How tokenization works

- **Tokenization** is the process of turning a meaningful piece of data, such as an account number, into a random string of characters called a token that has no meaningful value if breached
- Tokens serve as references to the original data, but cannot be used to reverse-engineer those values
- There is no key, or algorithm, that can be used to derive the original data from a token



All contents © MuleSoft Inc.

51

How tokenization works

- Tokenization uses a database, called a **token vault**
 - Stores the relationship between the sensitive value and the token
 - Policy name, operation (tokenization/detokenize), and data is part of the tokenization call
- The real data in the vault is then secured, often via encryption
- Anypoint Exchange has a connector for tokenization



All contents © MuleSoft Inc.

52

Anypoint Security also implements vaultless tokenization



- Distributed mapping tables shared at each node
- No vault or database is needed to manage or replicate across availability zones / Data centers



All contents © MuleSoft Inc.

53

Evaluating Mule application security risks



- **Threat** - Consider anyone who can send untrusted data to the Mule application, including users, internal users and administrators
- **Found in**
 - DB processors who accept query text or input parameter such as Execute Script, Execute DDL
 - DB processors accept query text and attackers are likely to send malicious data in query text which can alter data integrity, perform unintended DML or DDL statement on DB
 - XPath extract
 - Xpath extract can execute on unintended element in XML file which can provide sensitive data to attacker
 - Parse template
 - Parse template can provide access to secure properties on server if file system is not protected
 - Headers for HTTP
 - Malicious values in header can cause DoS

- **Prevention**
 - Use a Validation component or other defensive coding
 - Use bind variables in an SQL query instead of string manipulation
 - Avoid dynamic queries
 - Apply Schema validation using APIkit router or XML validate schema components

Mule application security risks - Broken authentication and session management



- **Threat** - Consider anyone who could steal accounts from others
- **Found in**
 - Authentication provider when user authentication credentials are not protected
 - Authentication provider is not storing credentials using hashing or encryption
 - Connectors when password, session ids, credentials are sent on unencrypted connections or embedded in URL
 - Credentials are sent in clear text in URL on non secure transport for connectors
 - Invalidate oauth context
 - OAuth sessions are not invalidated after logout

Mule application security risks - Broken authentication and session management



- **Prevention**
 - Store credentials using hashing or encryption
 - Use secure properties files for customer-hosted deployments
 - Use safely hidden properties for CloudHub deployments
 - Use secure connectors for communication
 - Invalidate sessions promptly
- Note: secure properties and safely-hidden properties are discussed later in this module

Mule application security risks - Security misconfiguration



- **Threat** - Consider anonymous attackers or users who want to compromise systems to gain unauthorized access
- **Found in**
 - ALL connectors
 - Credentials for connectors using default username and password
 - APIkit console
 - APIkit console access is left enabled for API
 - Heart beat or system information or utility apps
 - Heart beat or system information or utility apps are unprotected in application
 - Error handling
 - Error stack trace provide informative information to user

Mule application security risks - Security misconfiguration



- **Found in**
 - Unused message sources, connectors
 - Unused message sources can cause lead unintended usage of application resources
 - Unused ports on HTTP listeners can provide access to application endpoints and resources
- **Prevention**
 - Harden Mule applications and properly lock down the deployed environment
 - Use a strong architecture
 - Periodic security scans and audits
 - Standardized error handling providing error information relevant to the user without providing unneeded or unsecure information about the underlying system

Mule application security risks - Sensitive data exposure



- **Threat** - Consider anyone who can gain access to sensitive data at rest or transit
- **Found in**
 - Application properties
 - Attacker can read clear text files with system credentials which can cause any zero day attack for organization
 - Persistent store such as Object store or file
 - If Object store is not private for sensitive information, then attacker can retrieve sensitive data from Object store
 - Security certificate store
 - If security certificate store is not protected then attacker can have access to unintended services on network

Mule application security risks - Sensitive data exposure



- **Prevention**
 - Use secure properties
 - Use encryption, hashing, or masking for sensitive data (PII or PCI)
 - Protect access to resource using Anypoint Platform enterprise security or OS/VM level grants/permissions

Mule application security risks - Using component with known vulnerabilities



- **Threat** - Some vulnerable components and libraries can be identified and exploited with tools
- **Found in**
 - TLS for HTTP/S, SFTP, SMTP/S and Socket
 - TLS 1.0 is open to **man in middle attack** risking the integrity and authentication of data sent between client and server
 - Custom libraries
 - Untested libraries can exploit application, server, host, for example - Apache Struts vulnerability caused remote attacker to execute arbitrary code on any server running an application built using the Struts framework

Mule application security risks - Using component with known vulnerabilities



- **Prevention**
 - Use TLS 1.2/1.1 (Do not use deprecated TLS 1.0/SSL 3.0)
 - Use only approved enterprise libraries
 - Perform static and dynamic code analysis of Java sources or byte code

Exercise 15-2: Identify security threats exposed by a sample application



- Review and identify security threats exposed in a Mule application

Exercise steps



- Import sample Mule application (exercise-15-2.jar) in your student files
- Go to “security-misconfigurationFlow”
- Verify various components and property files in flow
- Check for below security threats
 - Security misconfiguration
 - Sensitive data exposure
 - Broken authentication and session management
- Identify threats exposed by flow in each category

- Security misconfiguration
 - Password, security key are in clear text
 - Unused/vulnerable properties in property file
 - Misconfiguration of properties
- Sensitive data exposure
 - Data stored in database in clear text
 - Data transmitted in clear text
- Broken authentication and session management
 - User authentication credentials are not protected
 - Credential can be guessed
 - Session did not timeout
 - Credential sent over unencrypted communication

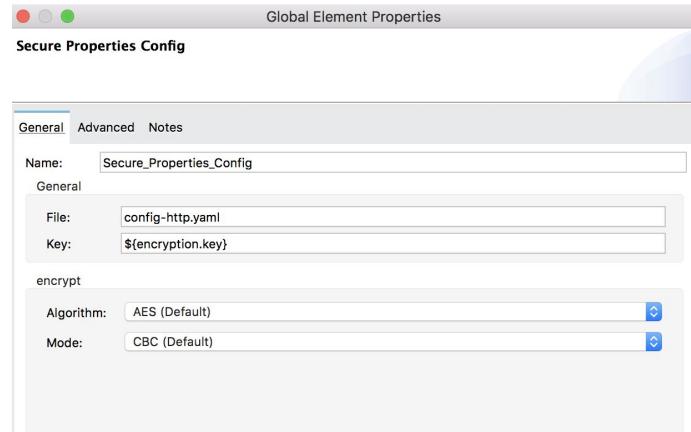
Securing application properties



Securing application properties



- Secure Properties Config encrypt properties and store the encrypted text in secure properties files
- Supports multiple encryption algorithms and mode
- Encryption key for encryption sets as system property or env variable
- Define one secure config for each secure property file
- Install extension as dependency in pom
- AES algorithm with CBC mode is default



All contents © MuleSoft Inc.

Define secure config



- Each secure property file needs one secure config

```
<secure-properties:config name="Secure_Properties_Config"  
file="config.yaml" key="${encryption.key}">  
    <secure-properties:encrypt algorithm="AES" mode="CBC"/>  
</secure-properties:config>
```

All contents © MuleSoft Inc.

- The **Secure Configuration Properties** module supports both YAML configuration files and Java properties files
- Encrypted value is defined in **![value]** in quotes

```
db:  
  host: "localhost"  
  port: "3306"  
  user: "root"  
  url: "training"  
  password: "![DJqeCF2q+gwuHPxNw6+apA==]"
```

- The **secure:: prefix** is used to access encrypted and clear-text values from secure property files

```
<db:config>  
  
  <db:my-sql-connection host="${secure::db.host}"  
    port="${secure::db.port}" user="${secure::db.user}"  
    password="${secure::db.password}"  
    database="${secure::db.url}" />  
  
</db:config>
```

How to update secure properties



- Can be updated at design or deployment time
- To override or update properties using the Runtime Manager console's property tab (or REST API), values are entered as plain text, NOT as an encrypted value
 - Runtime Manager cannot access the decryption key
 - CloudHub deployments can safely hide the new value

key	value
sqldb.password	mule
key	value

All contents © MuleSoft Inc.

73

Safely hiding application properties in CloudHub Mule application deployments



- CloudHub supports **safely hiding** Mule application properties
- Properties are encrypted and stored in the online CloudHub properties service (database)
 - The value can then never be seen in the Runtime Manager GUI

```
{ } mule-artifact.json X
1  {
2    "configs": [
3      "scalability-reliability-module.xml"
4    ],
5    "secureProperties": [
6      "secure::encryption.key"
7    ],
8    "redeploymentEnabled": true,
9    "name": "scalability-reliability-module",
10   "minMuleVersion": "4.1.2",
11   "requiredProduct": "MULE_EE",
12   "classLoaderModelLoaderDescriptor": {
13     "id": "mule",
14     "attributes": {
15       "exportedResources": []
16     }
17   },
18   "bundleDescriptorLoader": {
19     "id": "mule",
20     "attributes": {}
21   }
22 }
```

All contents © MuleSoft Inc.

74

- On-premises
 - Set from JDK system properties
 - Mule -M-Dencrytion.key=Mule
 - Set from environment variables from OS
 - varies with OS
 - Set from wrapper.conf in <Mule_HOME>/conf
 - wrapper.java.additional.<n>=-Dencrytion.key=Mule
- CloudHub
 - Set from Runtime Manager console's property tab
 - Best practice is to set key from Runtime Manager console and hide as application property by listing under secureProperties key as comma separated list in mule-artifact.json

Exercise 15-3: Prevent misconfiguration of secure properties for a Mule application

- Prevent the security misconfiguration identified in the sample Mule application exercise-15-2.jar in the solution of Exercise 15-2

Exercise steps



- In CloudHub, import exercise-15-3.jar for analysis
- Verify the following conditions:
 - Password, security key are in clear text
 - Unused/vulnerable properties in property file
 - Misconfiguration of properties

Exercise steps: How to prevent password, security key in clear text



- Choose Configuration Property or Secure Configuration Property
- Decide algorithm and mode
- Choose security key
- Verify the security key is plain text and readable
- Discuss ways to secure the security key

Exercise steps: How to prevent unused/vulnerable and misconfiguration properties



- Review the Mule application for unused/vulnerable properties in the property file
- Review the Mule application for misconfiguration of properties

Exercise solution: Define Security config



```
<secure-properties:config name="Secure_Properties_Config"  
    file="config-http.yaml" key="${encryption.key}" >  
    <secure-properties:encrypt algorithm="AES" mode="CBC"/>  
</secure-properties:config>
```

Exercise solution: How to secure key



- Analyse options to secure the key
 - Set property placeholder values through a JDK system property from the command line
 - Set from the wrapper.conf file
 - Set from an env variable in the operating system
 - Set from the Runtime Manager console's Property tab
- Choose an option to secure the key
 - CloudHub does not provide control over JDK system property, wrapper.conf or OS env variable and best option to secure key is set from Runtime Manager console's property tab

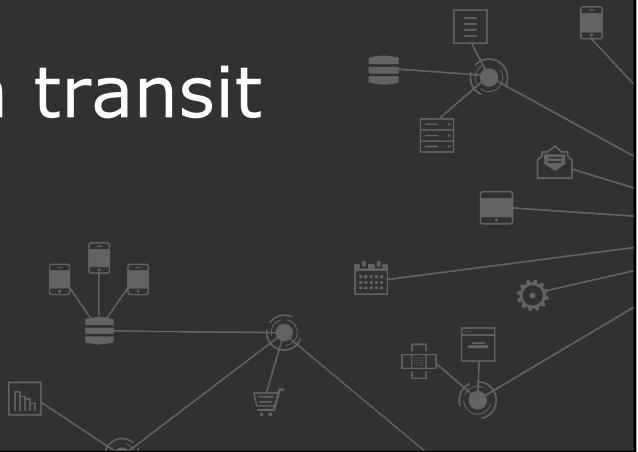
Exercise solution: Prevent security misconfiguration for sample application



```
db:  
  host: "localhost"  
  port: "3306"  
  user: "root"  
  url: "training"  
  password: "![DJqeCF2q+gwuHPxNw6+apA==]"
```

The screenshot shows the 'Properties' tab of the MuleSoft Anypoint Studio interface. A configuration entry for 'password' is displayed, with its value ('![DJqeCF2q+gwuHPxNw6+apA==]') shown in a redacted state, indicated by a series of dots (...).

Securing data in transit



How to secure data in transit



- Network related security responsibilities usually carried out by ops
 - Configuring network protocols to protect data in transit
 - HTTPS, SSL, TLS
 - Configuring Anypoint Edge security can also protect data in transit
 - These options are covered in the next module
- Security responsibilities usually carried out by Mule developers
 - Needing to encrypt data within the Mule application
 - Mule provides a Crypto security module to encrypt and decrypt data inside a Mule application

How developers secure data in Mule applications



- The Mule Cryptography module supports a Java Cryptographic architecture for
 - Symmetric or asymmetric encrypting/decrypting
 - Signing and signature validation of a payload or part of a payload
- Supports
 - Java cryptography extension (JCE)
 - Pretty good privacy (PGP)
 - XML
- DataWeave also has a crypto module for hashing payload
 - import dw::Crypto in DW to use Crypto functions

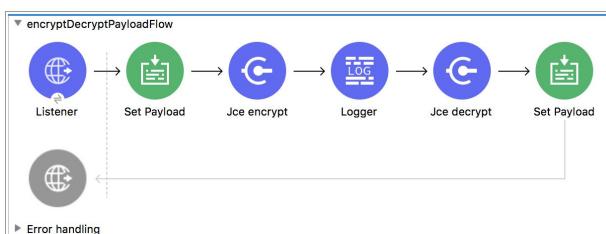
Search in Exchange..	Calculate checksum
+ Add Modules	Jce decrypt
★ Favorites	Jce decrypt pbe
Core	Jce encrypt
Crypto	Jce encrypt pbe
Database	Jce sign
Email	Jce sign pbe
File	Jce validate
HTTP	Jce validate pbe
Java	Pgp binary to armored
JMS	Pgp decrypt
OAuth	Pgp encrypt
ObjectStore	Pgp encrypt binary
Secure Properties	Pgp sign
SFTP	Pgp sign binary
Sockets	Pgp validate
Spring	Validate checksum
	Xml decrypt
	Xml encrypt
	Xml sign
	Xml validate

All contents © MuleSoft Inc.

How Crypto works



- For encrypting/decrypting and signing and signature validation of a payload
- To generate keystore file
- Java Keytool generate keystore file using
 - keytool -genkey -alias selfsign -keyalg AES -keystore aesEncryptionKey.jceks -keysize 128 -storeType JCEKS
- Mule Crypto processors use JCE configuration with generated key store file



Summarizing security best practices



Security best practices



- Use "Least privilege access" for Anypoint platform
- Do not rely on default security settings for platform or connectors
- Default error handling should be avoided as it provides lot of information to attackers
- Audit/logging should mask PII data
- Logging setting for application, runtime carefully set for environment

Summary



Summary



- Anypoint platform security provides various security features
 - Identity management
 - Client management
 - Mule secure property
 - Hiding application property
 - Mule Crypto module
 - DataWeave Crypto module
 - TLS for connectors
 - Anypoint Edge

- The customer needs to specify the level and type of security required by various integration solutions
- Anypoint Platform enables customers to build highly secure Mule applications that leverage enhanced inbuilt security in the platform
- The Mule runtime is built on top of standard Java security
- A Mule runtime can be configured to comply with the FIPS 140-2 standard

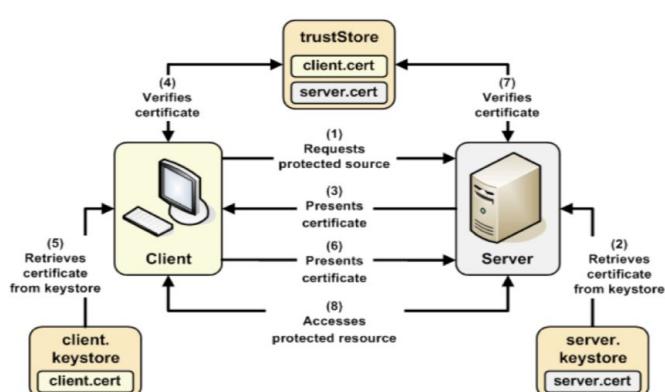
- Java SE Security
 - <https://www.oracle.com/technetwork/java/javase/tech/index-jsp-136007.html>
- Java Cryptography Architecture Standard Algorithm Name Documentation for JDK 8:
 - <https://docs.oracle.com/javase/8/docs/technotes/guides/security/StandardNames.html>
- Java Cryptography Architecture (JCA) Reference:
 - <https://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html>
- SSL for Apache Active MQ
 - <https://support.Mulesoft.com/s/article/How-to-enable-SSL-in-Apache-Active-MQ>



Module 16: Securing Network Communications between Mule Applications



Goal



TLS Context
Global TLS Configuration

General **Notes**

Name: Show password

Trust Store Configuration

Type: JKS
Path: clientStore.jks
Password: Show password
Algorithm:
Insecure: False (Default)

Key Store Configuration

Type: JKS
Path: serverStore.jks
Alias: selfsigned
Key Password: Show password
Password: Show password
Algorithm:

Advanced

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



- Configure secure communication between Mule applications and Mule runtimes
- Secure certificates in a Mule runtime
- Identify how network security works
- Configure a virtual private network (VPN)

Securing Mule application communications



How transport layer security (TLS) is used with Mule applications



- TLS **encrypts** data sent to and from Mule applications and other systems or applications
- **Certificates** are used for one-way or two-way handshakes
 - Certificates are used for **asymmetric public/private key** cryptography and to verify and trust the identity of a client, server, or other entity
 - In a one-way handshake the client must trust the server's identity
 - In a two-way handshake, the server must also trust the client's identity
 - Trust is often established by a chain of trusted certificate signing authorities
- Data **encrypted** with the **public** key can **only** be **decrypted** with the **private** key

Reviewing asymmetric cryptography



- Asymmetric cryptography uses a public/private key pair
 - A message encrypted with the public key can only be decrypted with the corresponding private key
 - A message encrypted (signed) by a private key can only be decrypted with the corresponding public key
 - The public key is usually exchanged with one or more other parties
- This is used to secure communications, such as with SSL or TLS



What is contained in a public certificate



- A **public certificate** contains credentials
 - The public key (but not the private key)
 - Organization details
 - The certificate issuer
- A certificate is typically signed by a trusted Certificate Authority (**CA**) or can be self signed using the Java keytool
 - Other people or systems can use the trusted CA's public key to validate the authenticity of the CA signature, and hence the certificate

How asymmetric cryptography is used to authenticate (trust) publicly available data



2. The server has a **certificate** (public key) that is signed by a trusted certificate authority (CA) using the CA's private key
 - The server sends its **signed** (encrypted) **certificate** to the client
 - The client uses the CA's public key to decrypt/**authenticate** the server's certificate
 - This guarantees the server certificate (public key) is **valid** and **untampered**
 - The client now **trusts** the server is the only entity with the private key corresponding to the **server** certificate's **public key**



How asymmetric cryptography is used to initiate a secure communication channel



3. The client **encrypts** a **symmetric key** into a message using the server **certificate** (public key), then **safely** sends the message to the server
4. Only the server has the **private key** to unlock the symmetric key in the message
 - No one "in the middle" can steal the symmetric key
 - Even the client cannot decrypt the message, but the client still has the original version
 - Now the client and server can safely communicate over the internet using the same symmetric key to encrypt and decrypt messages, which is faster



All contents © MuleSoft Inc.

10

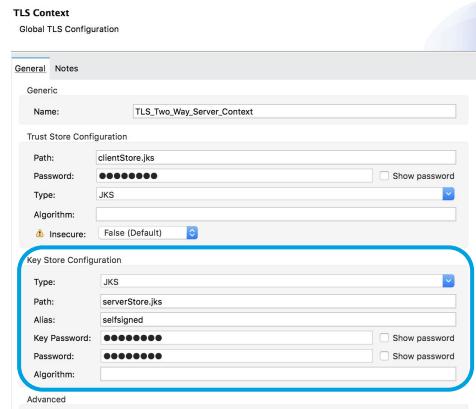
Securing Mule applications using Java key stores



Understanding Java keystores



- A **keystore** stores public certificates plus corresponding private keys (credential) for clients or servers in a Mule application
- The Java keytool can be used to create keystores for Mule applications or Anypoint Platform
- Public certificates in a keystore have a **certificate chain associated** with them, which **authenticates** the corresponding public key
- In TLS, the keystore determines the credentials (public certificate) sent to the remote host (e.g., client)



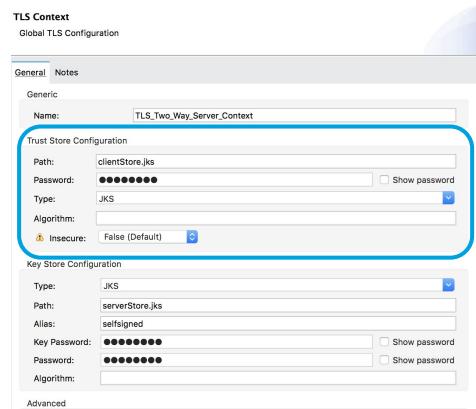
All contents © MuleSoft Inc.

12

How truststores are created and used Anypoint Platform and Mule applications



- The **Java keytool** is used to create **truststores**
- The **truststore** contains **public certificates** (self signed or from a CA) for remote hosts (other parties) and perhaps also the signing CAs



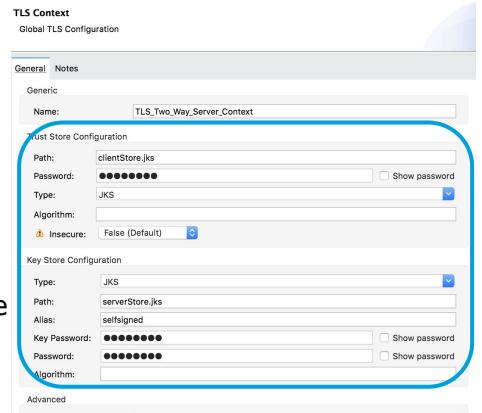
All contents © MuleSoft Inc.

13

Understanding Java truststores



- The **keystore owner** trusts the **public certificates** (and the contained public keys) contained in its truststore to identify other parties
 - Each other party must use the correct corresponding private key
 - Each private key is **not** stored in the truststore and is usually **not** available to the truststore owner
- In TLS, the truststore determines whether credentials (public certificates) sent by the remote host (the client) are trusted and hence if the secure connection can be established



How to configure the **insecure** parameter in a truststore



- The **insecure** parameter in a truststore element determines whether or not to validate the trust-store
- If set to true, no validation occurs
 - By default **insecure = false**
 - Setting **insecure = true** renders connections **vulnerable to attacks** and is recommended only for prototyping and testing purposes
 - For example, to troubleshoot security related issues, where the error and log messages are necessarily cryptic, turning off security features then incrementally adding them back can be helpful

Implementing TLS communications in Mule applications



Anypoint connectors that support TLS connections



- HTTP/S client and server
 - one-way or two-way TLS communication
- SFTP client
- SMTP/S clients
- TCP Sockets client and server
- JMS client
 - The JMS standard does not specify TLS connections
 - The JMS provider must support TLS and provide a TLS connection factory in its client library

Support for TLS in HTTP/S connector



One-way TLS

- The server sends its identity (public certificate containing its public key) to the client
- The client uses the server identity to safely exchange a symmetric key for private and encrypted two-way communication
 - Only the server can decrypt this key (man-in-the-middle attacks will fail)
- The server is not concerned (and ignores) the client's identity



18

Configuring one-way TLS for HTTPS



TLS Context
Global TLS Configuration

General Notes

Generic

Name:

Trust Store Configuration

Path:
Password: Show password
Type:
Algorithm:
Insecure: False (Default)

Key Store Configuration

Type:
Path:
Alias:
Key Password: Show password
Password: Show password
Algorithm:

Advanced

TLS Context
Global TLS Configuration

General Notes

Generic

Name:

Trust Store Configuration

Path:
Password:
Type:
Algorithm:
Insecure: False (Default)

Key Store Configuration

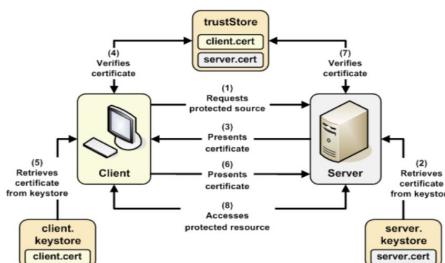
Type:
Path:
Alias:
Key Password: Show password
Password: Show password
Algorithm:

Support for two-way TLS in the HTTP/S connector



- Two-way TLS

- In this model, the client also sends its identity (public certificate, containing public key) to the server
- The client has its own keystore (with a private key)
- Optionally, may also include a truststore containing the server's public certificate
 - This is not required if the server certificate is signed by a well known CA
- Server has its own truststore to validate the client's certificate



20

Configuring two-way TLS for HTTPS



TLS Context
Global TLS Configuration

General Notes

Generic

Name:

Trust Store Configuration

Path:
Password: Show password
Type:
Algorithm:
Insecure: False (Default)

Key Store Configuration

Type:
Path:
Alias:
Key Password: Show password
Password: Show password
Algorithm:

Advanced

TLS Context
Global TLS Configuration

General Notes

Generic

Name:

Trust Store Configuration

Path:
Password: Show password
Type:
Algorithm:
Insecure: False (Default)

Key Store Configuration

Type:
Path:
Alias:
Key Password: Show password
Password: Show password
Algorithm:

Advanced

- A customer-hosted Mule runtime can externalize certificates to a folder outside the Mule_HOME location
 - Add a certificate folder in the classpath in the wrapper.conf file in <Mule_HOME>/conf
 - wrapper.java.classpath.3=%CERT_DIRECTORY%
 - Secure %CERT_DIRECTORY% using operating system permissions

Externalizing certificates in Mule runtime with Secret Manager

- **Secret Manager** can be used to store and control access to private keys, passwords, certificates, and other secrets
 - Currently, Secrets Manager is only supported for customer-hosted Mule runtimes in **Runtime Fabric** and **Anypoint Functional Monitoring**
 - On other unsupported deployment platforms (runtime planes) a custom certificate management solution could be used or created

Exercise 16-1: Identify transport layer security for a sample Mule application

- Identify transport layer security for a Mule application
- Identify ways to secure certificates for a Mule application

The image shows two side-by-side configurations of the 'TLS Context' screen in MuleSoft Anypoint Studio. Both configurations are titled 'Global TLS Configuration'.

Left Configuration (Client Context):

- General:** Name is set to 'TLS_Two_Way_Client_Context'.
- Trust Store Configuration:** Path is 'serverStore.jks', Password is '*****', Type is 'JKS', Algorithm is 'SHA256withRSA', and Insecure is 'False (Default)'.
- Key Store Configuration:** Type is 'JKS', Path is 'clientStore.jks', Alias is 'selfsigned', Key Password is '*****', Password is '*****', and Algorithm is 'SHA256withRSA'.

Right Configuration (Server Context):

- General:** Name is set to 'TLS_Two_Way_Server_Context'.
- Trust Store Configuration:** Path is 'clientStore.jks', Password is '*****', Type is 'JKS', Algorithm is 'SHA256withRSA', and Insecure is 'False (Default)'.
- Key Store Configuration:** Type is 'JKS', Path is 'serverStore.jks', Alias is 'selfsigned', Key Password is '*****', Password is '*****', and Algorithm is 'SHA256withRSA'.

Exercise steps

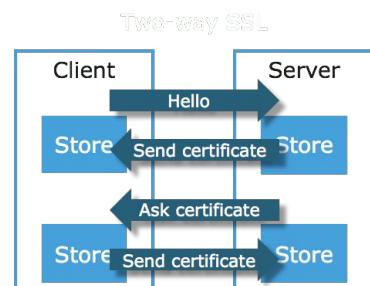
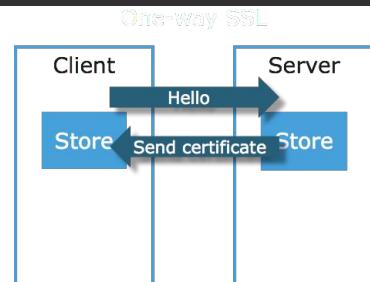
- Identify transport layer security for sample Mule application (security-misconfiguration.jar) of Exercise 16-1 deployed to a customer-hosted runtime plane
- Design ways to secure the Mule application's certificates

Exercise steps

- Identify clients of a Mule application
- Decide whether each client can secure private keys
- Decide the handshake mode between clients and servers (one-way or two-way)
- Identify mechanisms to secure server and client certificates

TLS best practices

- If you are authorizing a machine to access resources, it is advisable to have two-way TLS
- A customer-hosted Mule runtime should externalize certificates in a secure external directory



Reflection questions



- Do you use TLS in your Mule apps?
- If so, what issues did you have with TLS?

TLS Context
Global TLS Configuration

General Notes

Name: **TLS_Two_Way_Server_Context**

Trust Store Configuration

Type: clientStore.jks
Path: clientStore.jks
Password: **XXXXXXXXXX** Show password
Alias: selfsigned
Algorithm:

Insecure: False (Default)

Key Store Configuration

Type: JKS
Path: serverStore.jks
Alias: selfsigned
Key Password: **XXXXXXXXXX** Show password
Password: **XXXXXXXXXX** Show password
Algorithm:

Advanced

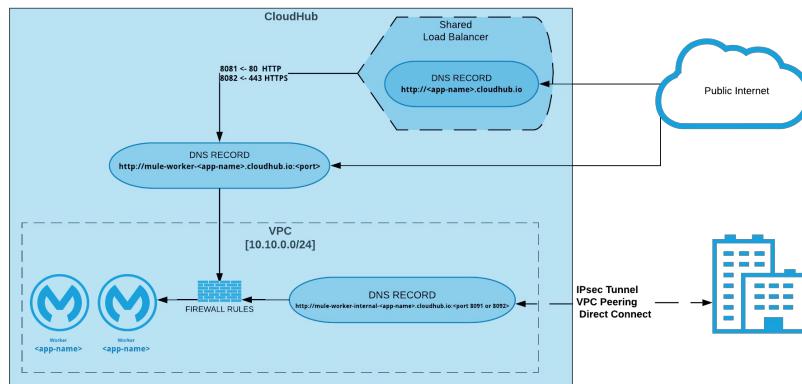
Designing network security options



Introducing Anypoint Virtual Private Cloud (VPC)



- Anypoint VPCs can create a private and isolated network in the cloud to host CloudHub workers
- Mule applications deployed to the VPC can communicate with each other using the VPCs private network addresses

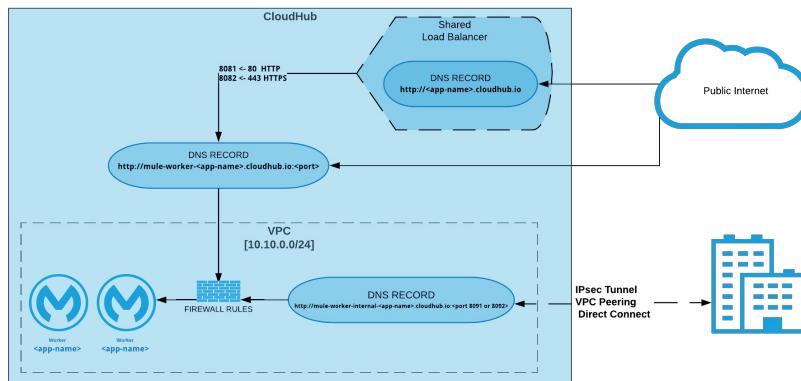


32

Using dedicated load balancers (DLBs) with a VPC



- The VPC can remain completely isolated from external networks
 - But usually a VPC uses one or more dedicated load balancers (DLBs) to route traffic with external public and private networks
 - A DLB can handle Mule event payloads up to 200 MB



33

Securing network communication through a CloudHub dedicated load balancer (DLB)



- CloudHub provides a shared load balancer for deployed Mule applications
- Dedicated load balancers can also be purchased and installed into a VPC
- Each DLB is provisioned in AWS and is tied to one CloudHub AWS region
 - Mule applications deployed to an AWS region and environment associated with a VPC and DLB can then receive traffic from the DLB
- Like the shared load balancer service, a DLB
 - Provides a DLB URL to load balance traffic to multiple workers

How a VPC isolated network is typically used

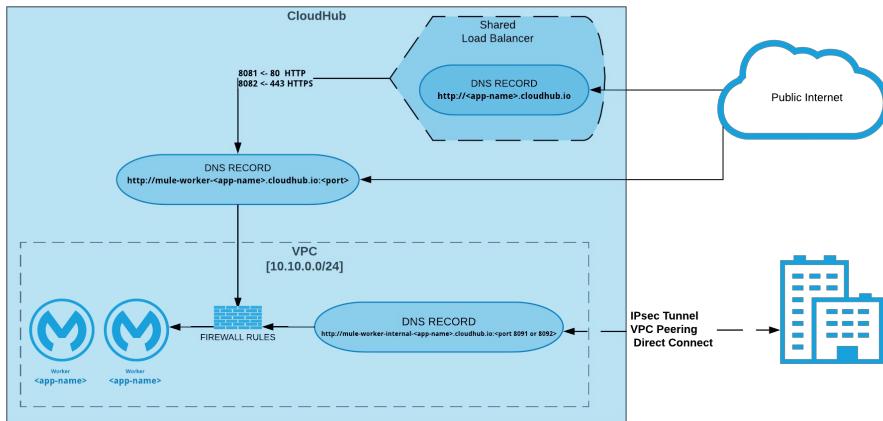


- Host your applications in a VPC and take advantage of DLBs
- Configure customized firewall rules for Mule applications deployed into the VPC
 - Open other ports besides 80, 8081, and 8082
 - Block direct connections to CloudHub workers over port 8081 or 8082
 - Allow or restrict any other TCP or UDP ports
- Connect a VPC to a corporate intranet
 - Whether on-premises or in other clouds
 - Via a VPN connection as if they were all part of a single, private network
- Connect your VPC to another AWS VPC using VPC Peering

Hiding the DLB hostnames



- A DLB's DNS entries are A records and are maintained by MuleSoft
- Customers can define their own "vanity domain names" as CNAMEs (for those A records) in their own DNS servers



36

Endpoint security using a CloudHub DLB



- A CloudHub DLB performs TLS termination
 - Just like the CloudHub Shared Load Balancer
- A CloudHub DLB must be configured with server-side certificates for a public-private key pair for the HTTPS endpoints it exposes
 - Optionally, client certificates can be added to a CloudHub DLB so that it performs TLS mutual (two-way) authentication
- The CloudHub DLB can enforce whitelisting of API clients
 - This is often used to define a CloudHub Dedicated Load Balancer for VPC-private APIs
- The DLB receives public IP addresses
- The public IP address' DNS name is an A record
 - So arbitrary CNAMEs for "vanity domain names" can be defined to point to that record

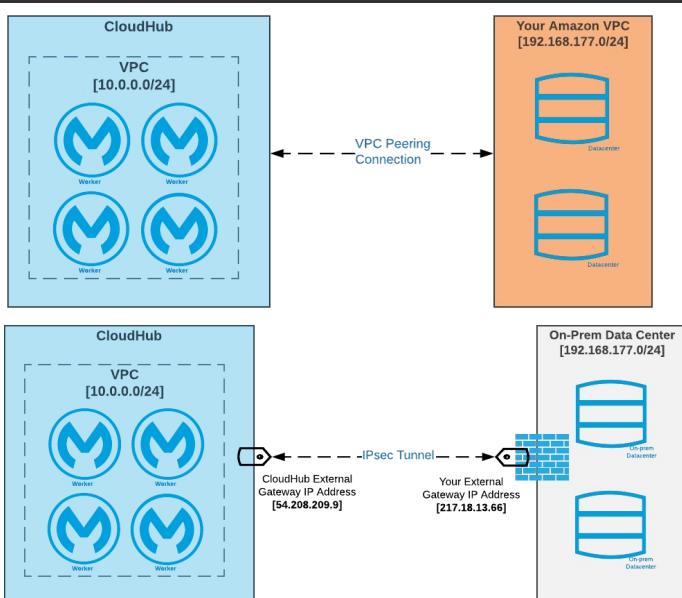
VPC connectivity methods

- Public internet
 - Default connectivity to CloudHub VPC
 - Less reliable as connectivity reliability drive by reliability of ISP
 - Moderate to no security as security is drive by endpoint of resource
- IPSec tunnel with network-to-network configuration
 - An Anypoint VPC can be connected to an on-premises network using an IPsec tunnel
- Amazon Direct Connect
 - An Anypoint VPC can be connected to an on-premises network using an AWS Direct Connect
- VPC Peering
 - Connect an Amazon VPC directly to a Anypoint VPC

All contents © MuleSoft Inc.

38

VPC peering vs VPN



39

- VPC peering uses **AWS infrastructure** and does not use the internet between regions
- Is the **preferred** way of communicating between an Anypoint VPC and a customer's AWS VPC
- Inter-region traffic is encrypted and intra-region traffic is not encrypted

Properly sizing a VPC to support Mule app deployments



- Mule applications are deployed in CloudHub workers and each worker is assigned with a dedicated IP
- For zero downtime deployment, each worker in CloudHub needs additional IP addresses
- A few IPs in a VPC are reserved for infrastructure (generally 2 IPs)
- The IP addresses are usually in a private range with a subnet block specifier, such as 10.0.0.1/24
- The **smallest** CIDR network subnet block you can assign for your VPC is /24 (256 IP addresses)
- The **largest** is /16 (65536 IP addresses)

Exercise 16-2: Decide an appropriate network subnet mask to properly size a VPC

- Decide an appropriate network subnet mask to properly size a VPC
 - Plan for two VPCs to split networking between production and non-production environments
 - Plan for the number of CloudHub workers required per anticipated Mule application
 - Calculate the minimum number of IP addresses required by each VPC
 - Plan a VPC future anticipated future growth

- The organization has **four** different environments
 - Dev, staging, performance, and production
- The organization is planning to deploy **50** applications in each environment
- The organization is planning to have **two** separate **VPCs** for production and non-production environments
- In the **performance** and **production** environments, each Mule application will be deployed to **two** workers
 - In other environments, the Mule application is only deployed to one worker
- Decide the **minimum** CIDR network subnet for the each of the VPCs (non-production and production environments)

Exercise context: Fill in this table

- Fill in this table to predict the minimum number of IPs required

Env	Production VPC	Non-production VPC
Dev		
Staging		
Performance		
Production		
Total		
Additional IP for zero downtime deployment(50%)		
Total IPs		

Exercise step



- Go to <https://www.ipaddressguide.com/cidr> and find the CIDR network subnet to support the minimum required total number of IPs

Exercise step



Env	Production VPC	Non-production VPC
Dev		50
Staging		50
Performance		$50 * 2 = 100$
Production	$50 * 2 = 100$	
Total	100	200
Additional IP for zero downtime deployment(50%)	50	100
Total IPs	150	300

- A subnet mask of **/23** will provide 512 IP addresses, which covers the required **minimum** number of 450 IP addresses
- However, remember that a VPC can be configured with any CIDR subnet between /24 and /16 **at no additional cost**
- Generally you should plan for **worst case expected growth**
 - If the VPC runs out of IP addresses, the VPC must be recreated and all Mule applications must be redeployed to the new VPC
 - The VPC should use the smallest subnet mask value corresponding to the largest number of IP addresses that Ops can support

Summary



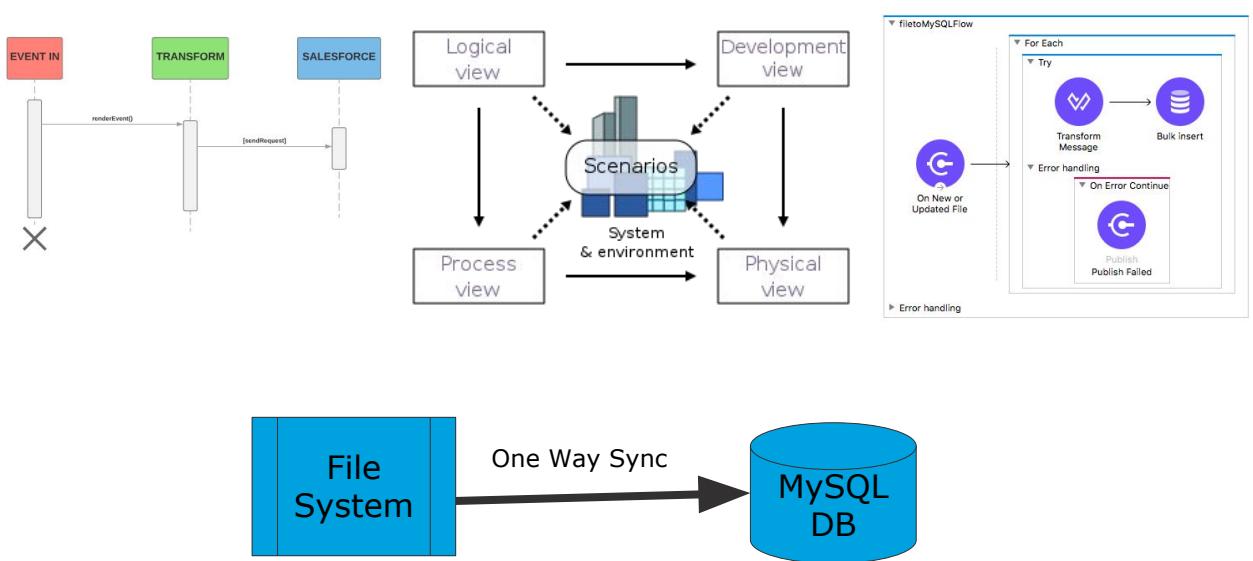
- TLS secures communications using asymmetric and symmetric cryptographic algorithms
- Asymmetric cryptography is more expensive performance-wise compared with symmetric cryptography
- Asymmetric cryptography is often used to exchange a symmetric key over the internet, or to sign certificates or other digital documents
- VPCs and dedicated load balancers can safely connect CloudHub environments with internal networks
- VPCs should be sized with enough IP addresses to accommodate expected growth



Module 17: Reviewing Documenting Integration Solutions Architectures



Goal



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



- Review the essential job tasks related to documenting integration solutions involving MuleSoft applications and Anypoint Platform
- Apply all the course job tasks to architect an integration solution architecture for a new use case



Designing and documenting integration solutions



Introducing the course wrap-up exercises



- The goal is to apply the essential job tasks discussed in this course to a new use case
- You will work independently to design and propose a solution, then the group will discuss the options and agree on a solution
 - You can mock flows in Anypoint Studio, or use any other architecture diagramming tool
- Each exercise iterates on your design to add additional details and complexity
 - Exercise 17-1: Appropriately design Mule event processing for a file transfer use case
 - Exercise 17-2: Respond to changing performance requirements
 - Exercise 17-3: Respond to changing reliability requirements

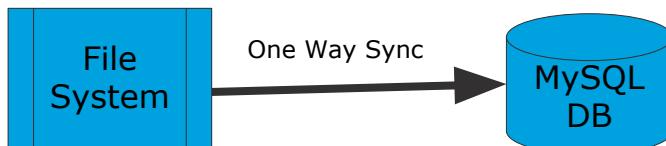
Essential job tasks related to documenting integration solutions



- In this class you have learned how to apply these essential job tasks to build a complete integration solutions architecture
 - Design integration solutions with Mule application flows and components
 - Apply appropriate event processing strategies in a Mule application to meet project requirements
 - Choose appropriate Mule event transformation and routing patterns
 - Choose appropriate state preservation and management options
 - Design secure Mule applications and network communications
 - Design testing strategies
 - Design effective logging and monitoring options
 - Decide and develop appropriate manual and automated deployment strategies
 - Design integration solutions architectures to balance high availability, reliability, transactionality, and performance goals

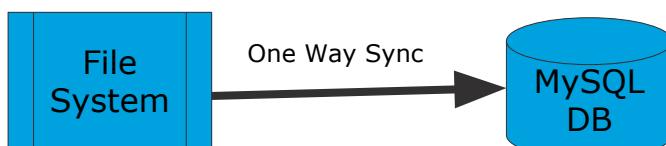
Exercise 17-1: Appropriately design Mule event processing for a file transfer use case

- Identify and recommend the Mule event processing models for a file transfer use case
- Identify and explain every factor that helps evaluate the best processing model
- Justify each Mule event processing model decision based on the identified factors
- Document flows that can implement the selected Mule event processing models



Exercise 17-1: Appropriately design Mule event processing for a file transfer use case

- Design for error handling and defensive programming
- Design for security requirements
- Design any needed state management and pick the best option
- Iterate on an integration solutions architecture to balance security, high availability, reliability, transactionality, and performance goals



- Requirements and constraints for the **File transfer** use case
 - Randomly, **a few times a day**, a **new source file** containing recent flight activity data for customers are **uploaded** to a specific **input** directory on the **file server**
 - The Mule application must quickly **process the file** and then send results to a **target MySQL database**
 - Each file has about **1000 records** of customer data
 - The Mule application has been budgeted to deploy to **one 0.2 vCore CloudHub worker** (with 1 GB of heap memory)
- Note: In this exercise, at least for your first solution, make performance goals less important than the other goals

- Requirements and constraints for the **Flight hold** use case
 - For each record in the received file, the requested flight is held
 - The customer sends requests to the flight API with an input payload structured like

```
{  
    "frequentFlyerId" : "TTT12345",  
    "destination" : "6778"  
}
```
 - First, the **destination** code is retrieved from the lookup service
 - Then flights for that destination code are retrieved from the American and Delta services
 - Then the two results are combined into a new data structure and sent to the database

Exercise steps



- Load the starter project from the Module 17 exercise_starter folder into Anypoint Studio
- Define options to process the file
 - What are the options to read in a file?
 - After reading in a file, how are records in files processed?
 - Should records be processed sequentially, parallelly, or in batch?
 - How can the memory footprint be reduced while processing?
 - How are failed records managed?

Exercise solution



- A proposed processing model using a For Each scope
 - Processing is sequential, even though records are published to a JMS topic
 - For Each does not support buffering (Mule 3) so the entire file is loaded into memory
 - Throughput of the process is determined and limited by the For Each scope
 - A separate insert into the DB is performed for each record

There are no errors.

Display Name: For Each

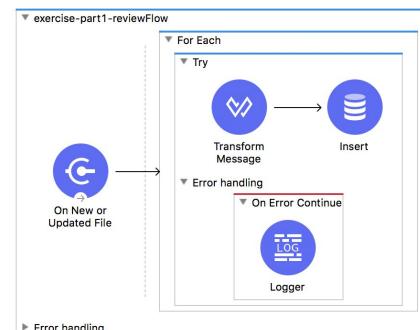
Settings

Collection:

Counter Variable Name: counter

Batch Size: 1

Root Message Variable Name: rootMessage



Exercise solution: Processing tradeoffs when using a For Each scope



Pros

- Auditing and tracing on records is easier
- Easier and isolated error handling of failed inserts to the target DB

Cons

- Throughput limited by the For Each scope

Exercise 17-2: Respond to changing performance requirements



- Modify an integration solution architecture to respond to new performance goals and other requirement changes

Exercise context: 6 months later... the file transfer use case requirements and constraints change



- The new source files containing recent flight activities for customers will still be uploaded at the same rate, **a few times a day**, to a specific **input** directory on the **file server**
- The size of each file has now **grown 100,000x** from 1000 records to **1M records** of customer data
- The Mule application must still quickly **process the file** and then send results to a **target MySQL database**
- The Mule application has been budgeted to deploy to **one 0.2 vCore CloudHub worker** (with 1 GB of heap memory)
- **Auditing** and **traceability** of each record is now deemed critical

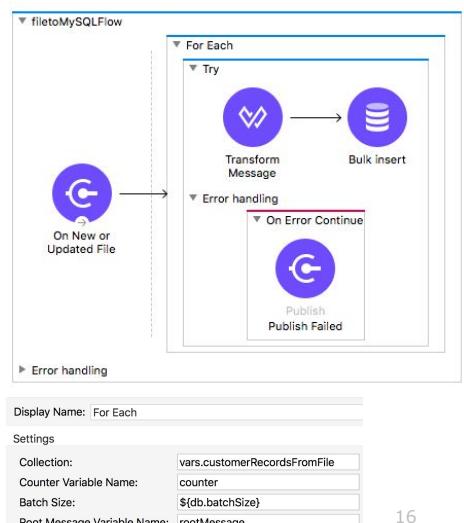
All contents © MuleSoft Inc.

15

Exercise solution



- A proposed processing model using a For Each scope configured to process the source file as streaming data
 - Each file is read by the flow as a **stream**
 - The For Each scope handles **batches** of records from the file
 - The batch size can be tuned by Ops as a property placeholder
 - In a Try Catch block, the stream is transformed using DataWeave
 - Records are then inserted into the target DB in a bulk insert using the same batch size set by the For Each scope
 - In the On Error scope, failed batches of records are sent to the DLQ



All contents © MuleSoft Inc.

16

Exercise solution : Factors drives to using a streaming



Factors	Optimum processing model
Large payload	<ul style="list-style-type: none">• Payload has 1 million records• Streaming is the best option for effective utilization of memory
Memory/CPU	<ul style="list-style-type: none">• Limited size of vCore requires the processing model should work with smaller memory and CPU footprints• Streaming is the best option

Exercise solution : Limitation when using a streaming

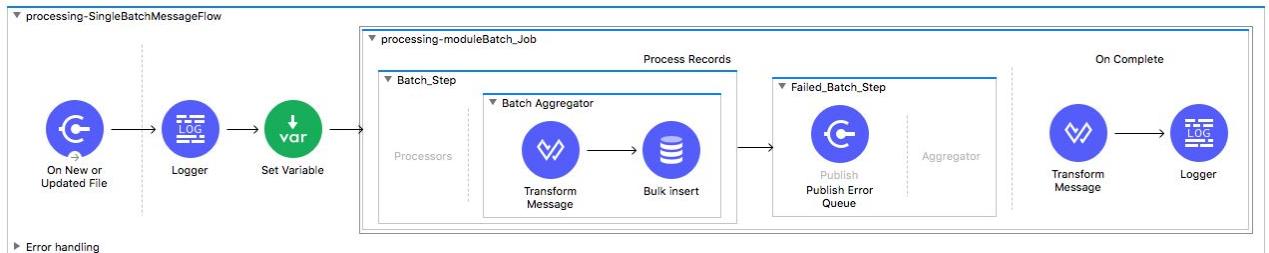


- **Auditing** and **traceability** of each record is not possible
- Traceability of failed records is limited
- **Human intervention** is required to **post process** any **failed records**

Exercise solution



- A different proposed processing model using a Batch Job scope
 - File is read as a stream in a Batch Job scope
 - The stream is transformed in a Batch Aggregator and then multiple records are inserted into the target DB in a single Bulk Insert operation
 - Failed records are sent to a JMS server to a dead letter queue (DLQ)



All contents © MuleSoft Inc.

19

Exercise solution: Processing tradeoffs when using a Batch Job scope

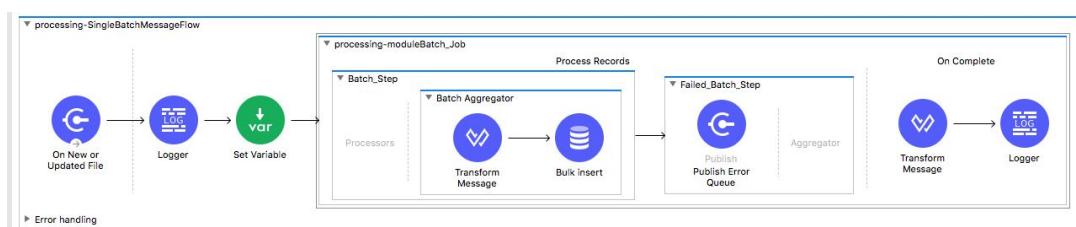


Pros

- Audit and tracing per record
- Auditing and tracing is easier with JMS
- Easier and isolated error handling of failed records

Cons

- Uses internal queues
 - May cause out of memory errors with large payloads and high throughput



All contents © MuleSoft Inc.

20

Exercise 17-3: Respond to changing reliability requirements



- Modify an integration solution to add real time data enrichment
- Modify an integration solution architecture to respond to new reliability SLA goals and other requirement changes

Exercise context: 6 months later... file transfer use case requirements and constraints change again



- A new requirement has been added to add additional flight details about a customers frequent flyer program to the target database records
- It has been observed that at random times the **server** to which the Mule application is deployed **shuts down while processing files**
 - This results in service outages and lost data
 - Some records have also been processed twice when the previous file is uploaded again
- You need to decide the best way to add additional **reliability** to the current implementation

Exercise steps

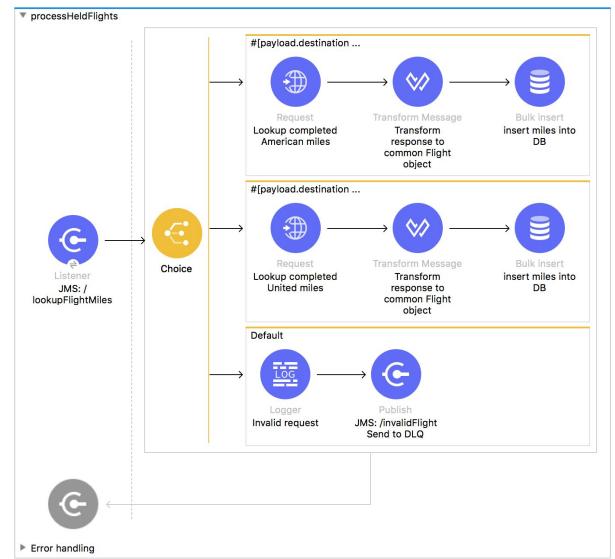
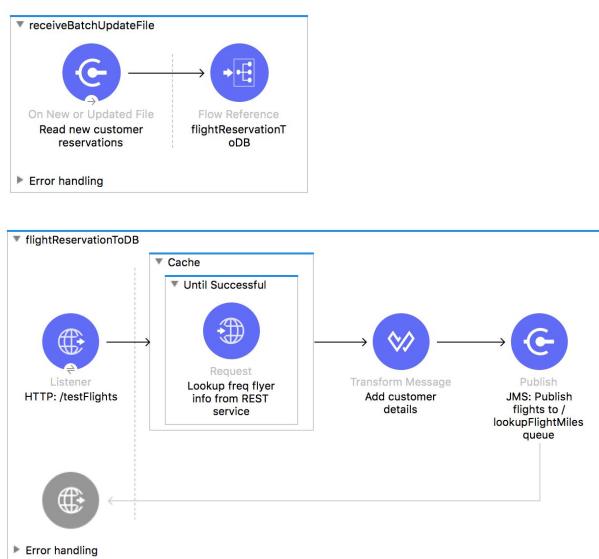


- Decide how to call out to existing web services to enrich the flight reservations received in the uploaded file
- Decide how to handle server crashes
 - When should processed files be deleted from the input directory of the file server?
 - Should processed files be moved to a different directory, and if so, when?
 - How can duplicate processing of the same data be avoided?

All contents © MuleSoft Inc.

23

Exercise solution



All contents © MuleSoft Inc.

24

- Can streaming help improve performance while still achieving reliability goals?
- What are the tradeoffs of batch vs. For Each vs. streaming?



Reviewing integration solution architectures using MuleSoft



Apply the course goals to a new use case



- Work with technical and non-technical stakeholders to translate **functional and non-functional requirements** into integration interfaces and implementations
- Create the **high-level design of an integration solution** and guide implementation teams on the choice of Mule Components and patterns to use in the detailed design and implementation
- Design Mule applications for any of the **available deployment options of the Anypoint Platform runtime plane**
- Apply standard development methods covering the full development lifecycle (project preparation, analysis, design, development, testing, deployment, and support) to ensure solution quality

27

Apply the course goals to a new use case



- Design reusable **assets, components, standards, frameworks**, and processes to support and facilitate API and integration projects
- Select the **deployment approach and configuration** of the Anypoint Platform with any of the available deployment options (MuleSoft-hosted or customer-hosted control plane and runtime plane)
- Design and be responsible for the **technical quality, governance** (ensuring compliance), and **operationalization** of the integration solution
- Advise technical teams on **performance, scalability, reliability, monitoring and other operational concerns** of the integration solution on Anypoint Platform

28

- Refine architecture views in integration architectures
- Document interfaces in integration architectures
- Document key assumptions, decisions, and tradeoffs in integration architectures
- Document NFRs and SLAs in integration architectures
- Document best practices in integration architectures

Refining architecture viewpoints for Mule applications

- **Deployment diagrams** are typically associated with use case realizations in the **Physical View** of the system
 - They document service and deployment models of the integration solution, including where and how MuleSoft tools and Anypoint Platform are used
- Sequence diagrams are typically associated with use case realizations of the Logical View of the system
 - They document the exchange of information across different systems and stakeholders
- Data transformation models describe how to connect external systems together
 - May involve common data models (CDMs) to promote reuse and decouple systems

- Documentation includes
 - Name and version the interface
 - Provide details of operations and their semantics
 - MuleSoft provides API design center to manage the API lifecycle
 - Provide what variances are available with respect to consuming the interface
 - Provide expected error conditions and error handling details
 - Provide performance or reliability numbers
 - Provide behavior diagrams like "sequence diagrams" in case the interaction is complex
 - Document dependency with external system
 - Use template for interface documentation
 - included in student files in starter resources folder in module 3

- Document decisions weighing in the different concerns and tradeoffs
- Keep a log of key decisions with details and audit trail
 - **Context or background**
 - Explain what the issue is about and the options that are available.
 - **Assumptions**
 - This includes the assumptions that are taken in context
 - **Decision**
 - This includes the decision that is taken and the rationale.
 - **Status**
 - This involves whether the decision is proposed or accepted.
 - There are various lifecycle events for a decision
 - **Impact**
 - What is the impact of the decision?
 - What do we gain or lose and what are the tradeoffs?
 - **Stakeholders**
 - Parties who are impacted by decisions

- The design approach should consider the nonfunctional requirements and related cost
- Various factors like **performance compliance, PCI and governance requirements**, etc. that impact the design are visible to the different stakeholders
- Different architectural components providing a service have SLA defined in their interface documentation
- The NFRs should show how different nonfunctional requirements are satisfied
- It helps different stakeholders like quality engineers and operational engineers to plan in advance various tasks like load testing, operational alerts, etc.

- Use some sort of standard templating around how the architecture document, interface design document should be produced
- Promote reusability of assets from Anypoint Exchange
- Document library and shared resources
- Standardized CI/CD for enterprise integration applications

Exercise 17-4: Document the architecture for the new use case



- Document the architecture for a new use case

Exercise steps



- Use your mocked design from the previous exercises to fill in an architecture document
- Use the starter template doc in the Module 17 starter folder
- Begin filling in sections of the architecture document
- Review a completed architecture solution with the instructor and discuss the decisions with the group

Summary



Summary

- Integration solutions are built in phases
- There are tradeoffs to decide options in all Mule application lifecycle stages
 - Design patterns
 - Defensive programming and error handling
 - Concurrent and parallel processing options
 - Reliability, high availability, and performance decisions and tradeoffs
 - Other non-functional requirements and SLAs
 - Runtime and control plane choices
 - Security options



Wrapping Up the Course



In this module, you will



- Review the **class objectives**
- Take the **class survey**
- Learn **where to go from here**
- Learn how to get **certified!**

Review the course objectives



Course goals are aligned with the corresponding MuleSoft certification exam



- The course goals are closely aligned with the job tasks for the **Mule Certified Integration Architect - Level 1 certification exam**
 - You will get an exam voucher after completing this class
<https://training.mulesoft.com/exam/mcia-level1>

The screenshot shows the MuleSoft Learning Platform interface. At the top, there is a navigation bar with links for Product, Solutions, Services, Resources, Company, and a search bar. Below the navigation bar is a sidebar containing a list of course modules, each with a downward arrow indicating more content. The modules listed are:

- Configuring and Managing Anypoint Platform
- Selecting Integration Styles
- Designing and Documenting Enterprise Integration Architecture
- Architecting Resilient and Performant Integration Solutions
- Handling Events and Messages
- Designing Applications with Anypoint Connectors
- Designing Networks for Anypoint Connectors
- Handling Integration Implementation Lifecycles
- Implementing DevOps
- Operating and Monitoring Integration Solutions

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



- Work with **technical** and **non-technical** stakeholders to translate **functional** and **non-functional requirements** into well documented **integration interfaces** and **detailed implementation designs**
- Guide implementation teams on the choice of **Mule Components** and **patterns** to use in the **implementation of integration solutions designs**
- Design reusable **assets, components, standards, frameworks, and processes** to support and facilitate API and integration projects
- Apply standard development methods covering the **full development lifecycle** (project preparation, analysis, design, development, testing, deployment, and support) to **ensure solution quality**

At the end of this course, you should also be able to



- Design Mule applications for any of the available Anypoint Platform **runtime planes**
- Select the **deployment approach** and **configuration of Anypoint Platform** with any of the available **deployment options** (MuleSoft-hosted or customer-hosted control plane and runtime plane)
- Design and be responsible for the **technical quality, governance** (ensuring compliance), and **operationalization** of the integration solution
- Advise technical teams on **performance, scalability, reliability, monitoring** and **other operational concerns** of the integration solution on Anypoint Platform

Take the class survey



Class survey



- You should have **received an email** with a link to the class survey
 - Your instructor can also provide the **direct link**
 - <http://training.mulesoft.com/survey/{surveyID}.html>
 - Or you can go to a general page and select your class
 - <http://training.mulesoft.com/survey/survey.html>
- **Please fill out the survey now!**
 - **We want your feedback!**
 - **Comments are especially helpful!**



Where to go from here



Take additional MuleSoft training courses



- Anypoint Platform **Architecture**:
 - Application Networks
- Anypoint Platform **Operations**:
 - CloudHub
 - Customer-Hosted Runtimes
 - API Management
- Anypoint Platform **Development**:
 - Mule 4 for Mule 3 Users
 - Advanced (Mule 3)
 - DataWeave (Mule 3)
- **Anypoint Platform**:
 - Flow Design
 - API Design



training.mulesoft.com

Get certified!



MuleSoft Certification program



- Offers multiple types of **professional accreditation** for developers and architects
- **Why get certified?**
 - To confirm you have mastery of the knowledge and skills required to perform your job role and tasks
 - To obtain recognized industry certification
 - To differentiate yourself in the marketplace
- **With each certification, you get**
 - A digital badge for your communications
 - The ability to add the certification to your LinkedIn profile



- **MuleSoft Certified Integration Architect - Level 1**

- Should be able to drive and be responsible for an organization's Anypoint Platform implementation and the technical quality, governance (ensuring compliance), and operationalization of the integration solutions
- The exam validates that an architect has the required knowledge and skills to work with technical and non-technical stakeholders to translate functional and non-functional requirements into integration interfaces and implementations



- **MuleSoft Certified Platform Architect - Level 1**

- Should be able to define and be responsible for an organization's Anypoint Platform strategy
- The exam validates that an architect has the required knowledge and skills to direct the emergence of an effective application network out of individual integration solutions following API-led connectivity across an organization using Anypoint Platform



MuleSoft Certified Integration Architect - Level 1 exam

- **Is proctored, closed book, 120 minutes, 58 multiple-choice**

- You can take exam at a testing center or online using your laptop camera



- **This class contains a voucher for two attempts for the exam!**

- You should have received an **email today** with a voucher code and instructions to take the exam
 - If you can't find the email, check your spam folder or filter



- Make sure you have mastered the content in this course before taking the exam
 - Review the **student slides**
 - Step through the **student manual** on your own
- Review the **exam topics**
 - training.mulesoft.com/exam/mcia-level1
- Take the **practice quiz**
 - training.mulesoft.com/mcia-level1-quiz



Thank You!