



**MuleSource**

the open source choice for SOA infrastructure

# **Integrating with Mule ESB**

**Ken Yagen, Senior Director of Engineering, MuleSource**

**[ken.yagen@mulesource.com](mailto:ken.yagen@mulesource.com)**

## ► MuleSource

- Mule project started in 2003
- MuleSource founded in 2006
- 1.5 million Mule downloads
- 2000+ production customers
- 8 of Fortune 50 customers

## ► Who am I?

- Responsible for development and delivery of MuleSource products
- 15 years of software development in enterprise (JP Morgan), large ISVs (BEA), and startups
- Background in enterprise-class middleware, security, and governance software

- ▶ Know what Mule ESB is and what it can do for you
- ▶ Learn how to build and consume services using Mule
- ▶ Learn how Mule can integrate cloud-based technologies and services
- ▶ Demo how to build an application using Mule
  - Demonstrate by Example
  - Best practices along the way
  - How to test your application



**MuleSource**

the open source choice for SOA infrastructure

# What is Mule?

- ▶ Making applications work together to achieve a purpose
- ▶ Why do we integrate?
  - Users don't care where business functions and information reside
  - Users expect instant access
  - Users want to access it through their browsers and mobile devices
  - Needs change faster than new software can be developed
  - Cheaper than building new solutions from scratch
  - We don't always own or control the pieces
- ▶ How do we integrate?
  - Historically, often point to point or through a centralized broker
  - Identify your systems and processes
  - Map data flows then model (EAI Patterns)
  - Define message formats

- ▶ SOA is the only thing Chuck Norris can't kill.
- ▶ SOA is not complex. You are just dumb.
- ▶ One person successfully described SOA completely, and immediately died.
- ▶ SOA can write and compile itself.
- ▶ SOA is being used in the developing world to solve hunger. Entire populations will be fed on future business value.
- ▶ In a battle between a ninja and a jedi, SOA would win.
- ▶ SOA violates the first and third laws of thermodynamics. But not the second, as all energy flows from SOA.
- ▶ The first rule of SOA is you do not talk about SOA.
- ▶ SOA in a Nutshell is 7,351 pages spread over 10 volumes.

- ▶ SOA is a business architecture
  - The practice of sequestering core business functions into independent services that don't change frequently (<http://objectmentor.com>)
- ▶ ESB is an architecture construct that can enable SOA
  - Event Driven, Standards Based, Messaging
- ▶ True SOA is disruptive
  - Requires changing how people think
  - Re-architecting existing systems is almost always necessary
  - Monolithic systems are replaced by loosely coupled, composite services
- ▶ But sometimes just need to integrate an existing business function
  - Service-orientation encourages good design
  - ESB can provide service abstraction and loose coupling
  - Better alternative to PTP integration

<b>JAVA</b>	1.4 / 5 / 6 / 7
<b>APPSERVER</b>	<ul style="list-style-type: none"> <li>▪ Apache Tomcat</li> <li>▪ Geronimo</li> <li>▪ Jetty</li> <li>▪ WebLogic</li> <li>▪ JBoss</li> <li>▪ JRun</li> <li>▪ WebSphere</li> <li>▪ Oracle</li> <li>▪ Resin</li> </ul>
<b>TRANSPORT</b>	<ul style="list-style-type: none"> <li>▪ JMS</li> <li>▪ MQ Series</li> <li>▪ File</li> <li>▪ FTP</li> <li>▪ HTTP</li> <li>▪ HTTP Servlets</li> <li>▪ HTTPS</li> <li>▪ IMAP</li> <li>▪ In-Memory</li> <li>▪ JBI</li> <li>▪ JDBC</li> <li>▪ SOAP</li> <li>▪ SSL</li> <li>▪ Multicast</li> <li>▪ Oracle AQ</li> <li>▪ POP3</li> <li>▪ Remote EJB</li> <li>▪ RMI</li> <li>▪ SMTP</li> <li>▪ System I/O</li> <li>▪ TCP</li> <li>▪ TIBCO</li> <li>▪ TLS</li> <li>▪ VFS</li> <li>▪ UDP</li> <li>▪ XMPP</li> <li>▪ AS400 Data Queues</li> <li>▪ File system</li> </ul>
<b>INTEGRATION</b>	<ul style="list-style-type: none"> <li>▪ Spring</li> <li>▪ EJB</li> <li>▪ GigaSpaces</li> <li>▪ JavaSpaces</li> <li>▪ JBI</li> <li>▪ JCA</li> <li>▪ JNDI</li> <li>▪ JOTM</li> <li>▪ JTA</li> <li>▪ PicoContainer</li> <li>▪ Plexus</li> <li>▪ HiveMind</li> </ul>
<b>WEB SERVICES</b>	<ul style="list-style-type: none"> <li>▪ XFire</li> <li>▪ Axis</li> <li>▪ SOAP</li> <li>▪ REST</li> <li>▪ Glue</li> </ul>
<b>SECURITY</b>	<ul style="list-style-type: none"> <li>▪ Acegi</li> <li>▪ JAAS</li> <li>▪ PGP</li> </ul>
<b>OTHER</b>	<ul style="list-style-type: none"> <li>▪ BPEL</li> <li>▪ JBPM</li> <li>▪ JSR-223 (Scripting)</li> <li>▪ OGNL Filters</li> <li>▪ Quartz</li> </ul>

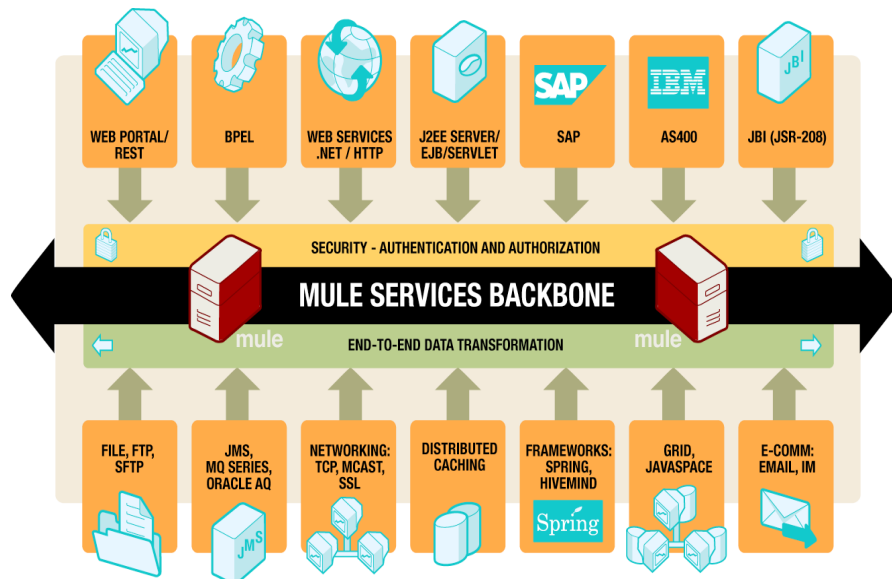
- ▶ Decouples Business Logic
- ▶ Location Transparency
- ▶ Transport Protocol Conversion
- ▶ Message Transformation
- ▶ Message Routing
- ▶ Message Enhancement
- ▶ Reliability (Transactions)
- ▶ Security
- ▶ Scalability

## ► Integration Style

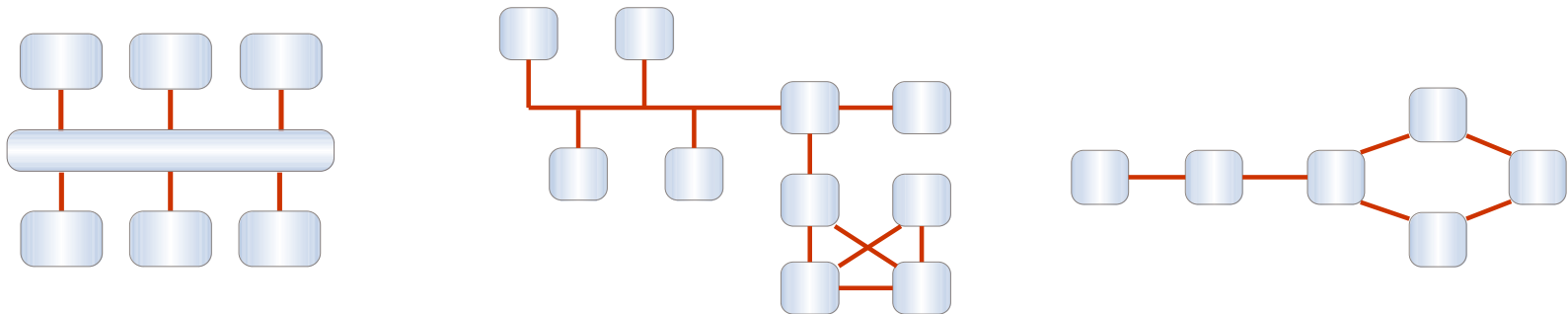
- Batch
- File Transfer
- Shared Database
- Request/Reply
- Messaging

## ► ESB advantages

- Supports all styles and bridges legacy systems
- Modular architecture
- Simple and flexible
- Easy to test and maintain
- Scalable
- Can be a step towards SOA

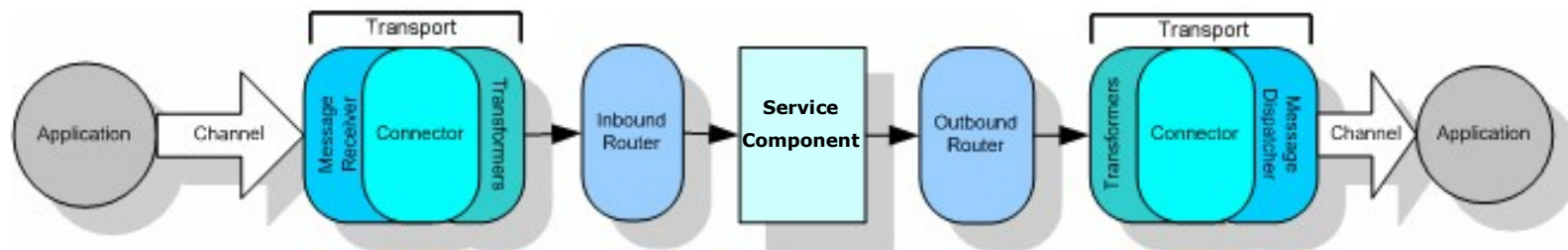
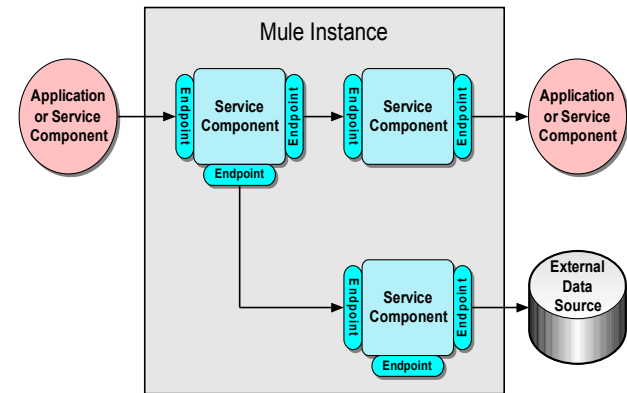


- ▶ World is full of events
- ▶ Run time composition through channels.
- ▶ Distributed processing nodes
- ▶ Loose coupling
  - Asynchronous Messaging
- ▶ More complexity in error handling, diagnostics
- ▶ Multiple Topologies
  - ESB, Peer-to-Peer, ESN, Pipeline, Client/Server, Hub & Spoke



► A Mule Instance includes set of key elements:

- Message
- Channel/Endpoint
- Transport
- Transformer
- Router
- Service Component





**MuleSource**

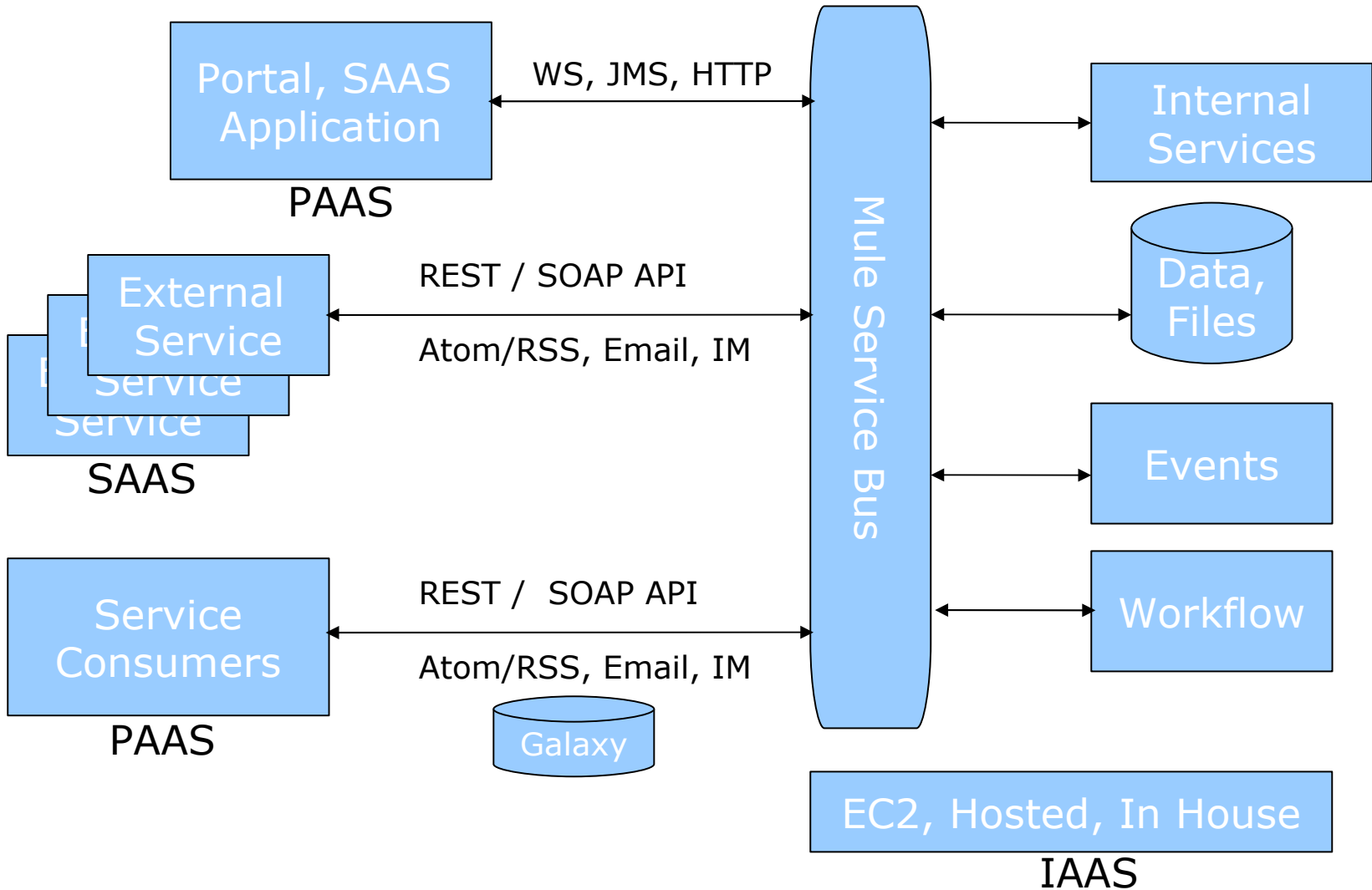
the open source choice for SOA infrastructure

# Integrating the Cloud

- ▶ Builds on SOA, Web 2.0, Virtualization, Grid Computing
- ▶ Offers new types of reusable services
  - Infrastructure (IAAS) – EC2, S3
  - Application Stack (PAAS) – SF, Google AppEngine
  - Business Services (SAAS) – Billing, CRM, Payment, Advertising
  - Utility Services - Communication, Data Processing, Doc Mgmt, Security
  - Information Services – Real Time Data, Historical Data, Search
- ▶ Low starting costs / Pay as you go / Massive Scalability
- ▶ Some Limitations
  - Immature SLAs, Customer Service
  - Security, Reliability, Management, Governance
  - Standard protocols but proprietary APIs and data formats
- ▶ Service-orientation critical to successfully developing cloud services and web mashups

- ▶ Expose and consume any type of service
  - Web Protocols: SOAP, HTTP/REST, RSS, ATOM, etc
  - Data formats: XML, EDI, JSON, Binary, Text
  - Service can POJOs, Spring, EJBs, Groovy, Python, Ruby, .NET, etc.
  - Service Registry (Galaxy)
- ▶ Mule manages complexity
  - Validate, transform, enhance data
  - Security, error handling, reliability and availability
  - Bridge internal and external services
  - Improves operational response time to issues
- ▶ Abstract Popular Cloud Services
  - Create a custom transport (eg SF Transport)
  - Configure endpoints rather than code to APIs
- ▶ Mule's lightweight approach
  - Simple deployment model
  - Standalone or embed in Tomcat

# Cloud Integration Example





**MuleSource**

the open source choice for SOA infrastructure

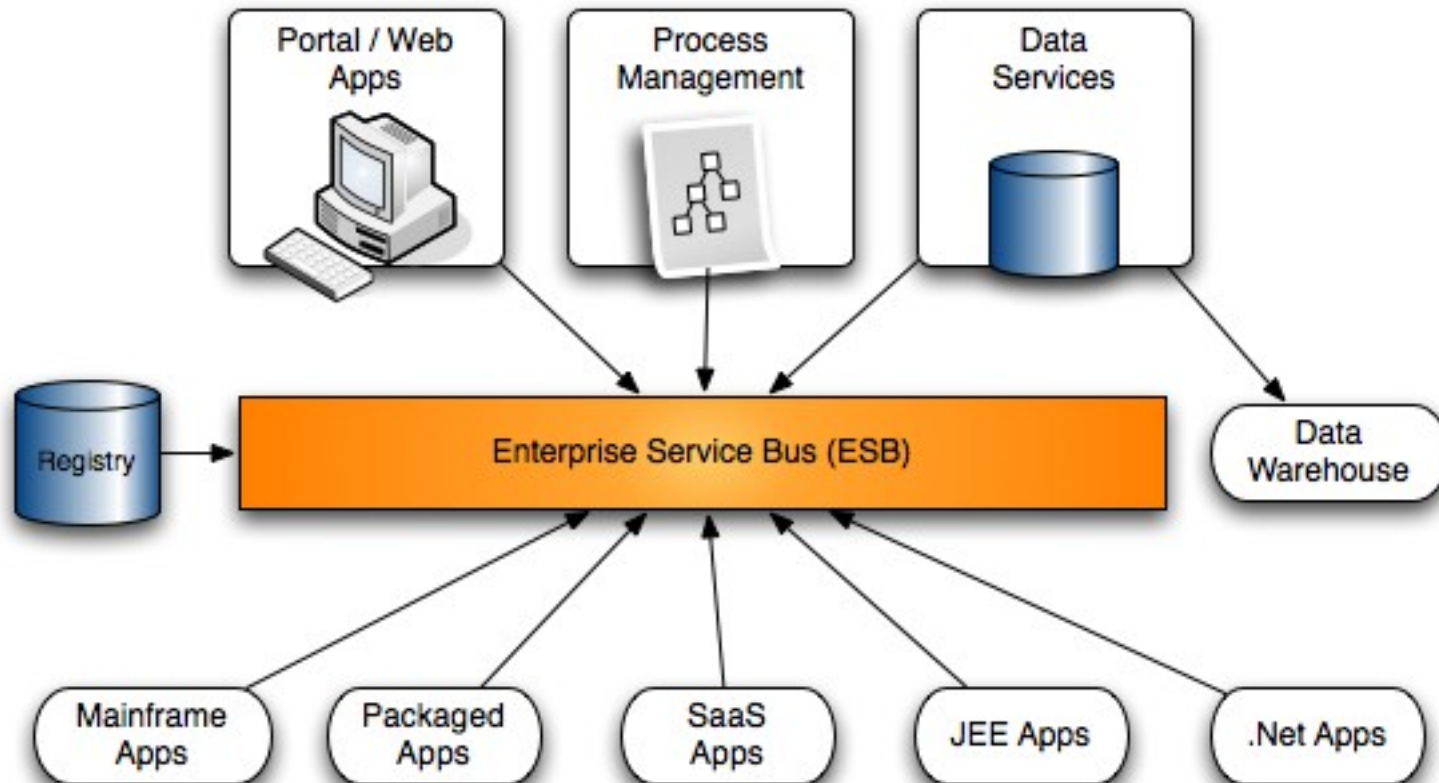
# Learning by Example

### Schnell Laptops

A boutique laptop distributor that sell high-end laptops. They have a growing online business, but need to make their back office processes more efficient so that they ship orders faster.

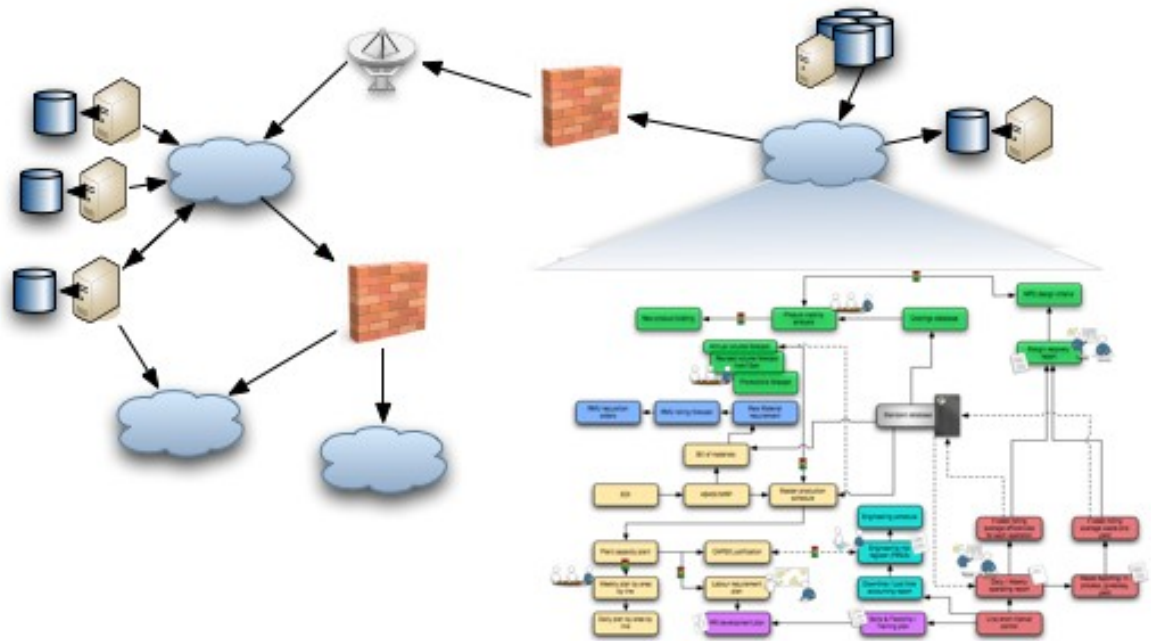
Currently, they have a lot of in-house code, point-to-point integrations, and manual operations, it's becoming difficult to change systems without breaking others.

## Vision

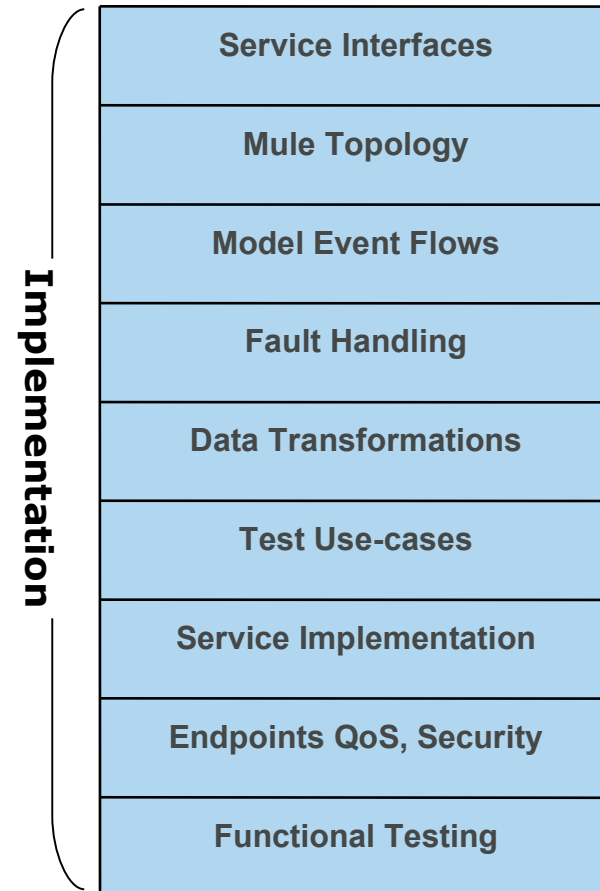
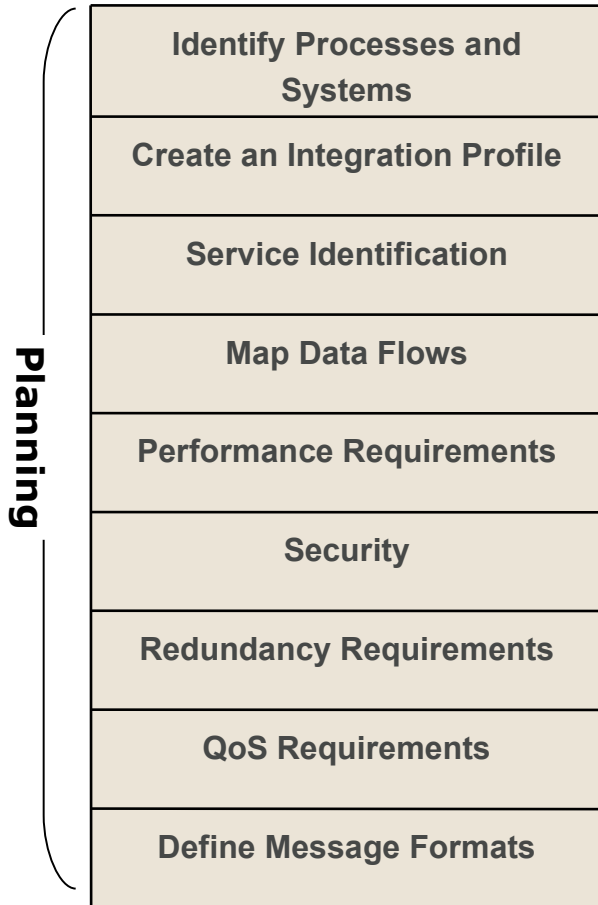


## Current Systems Architecture

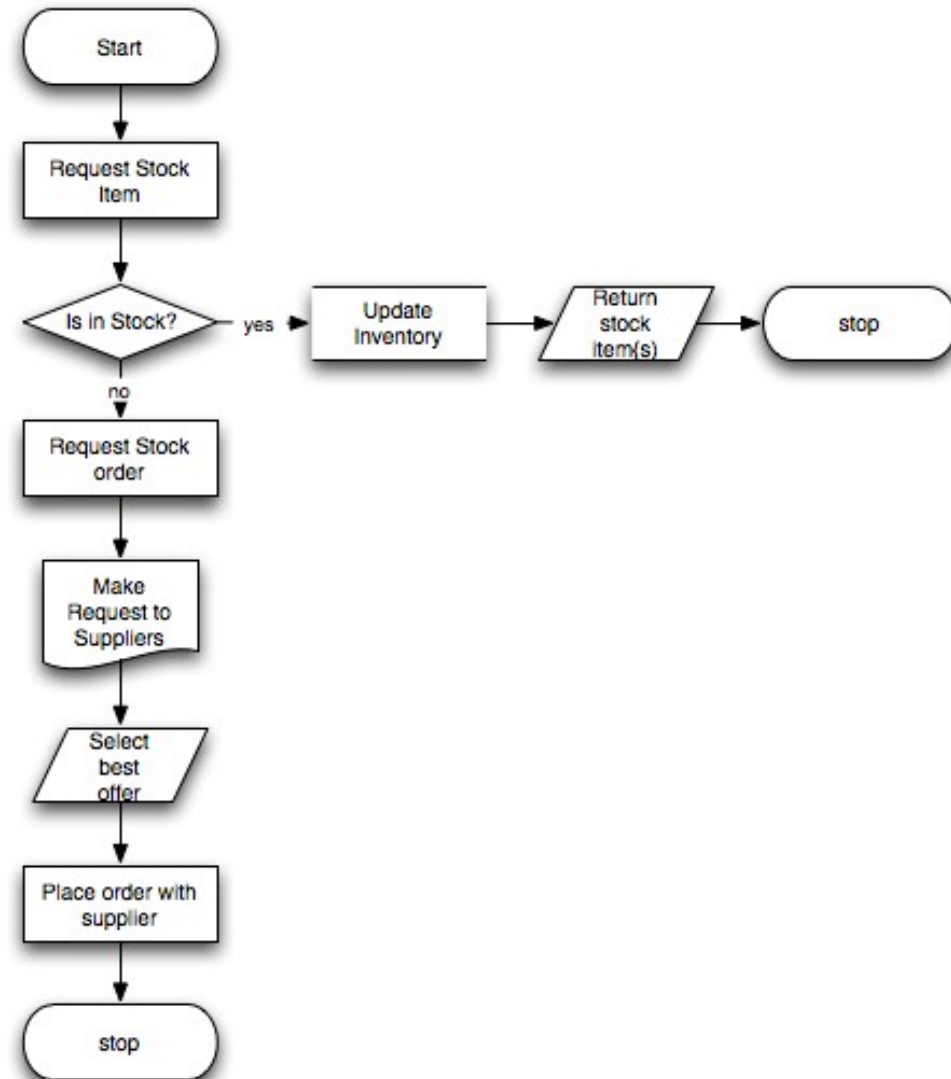
- Point to Point
- Home grown messaging abstraction
- RPC Only
- Hand coded 3<sup>rd</sup> party integration
- Hand coded security, monitoring, management








# Integration Check List

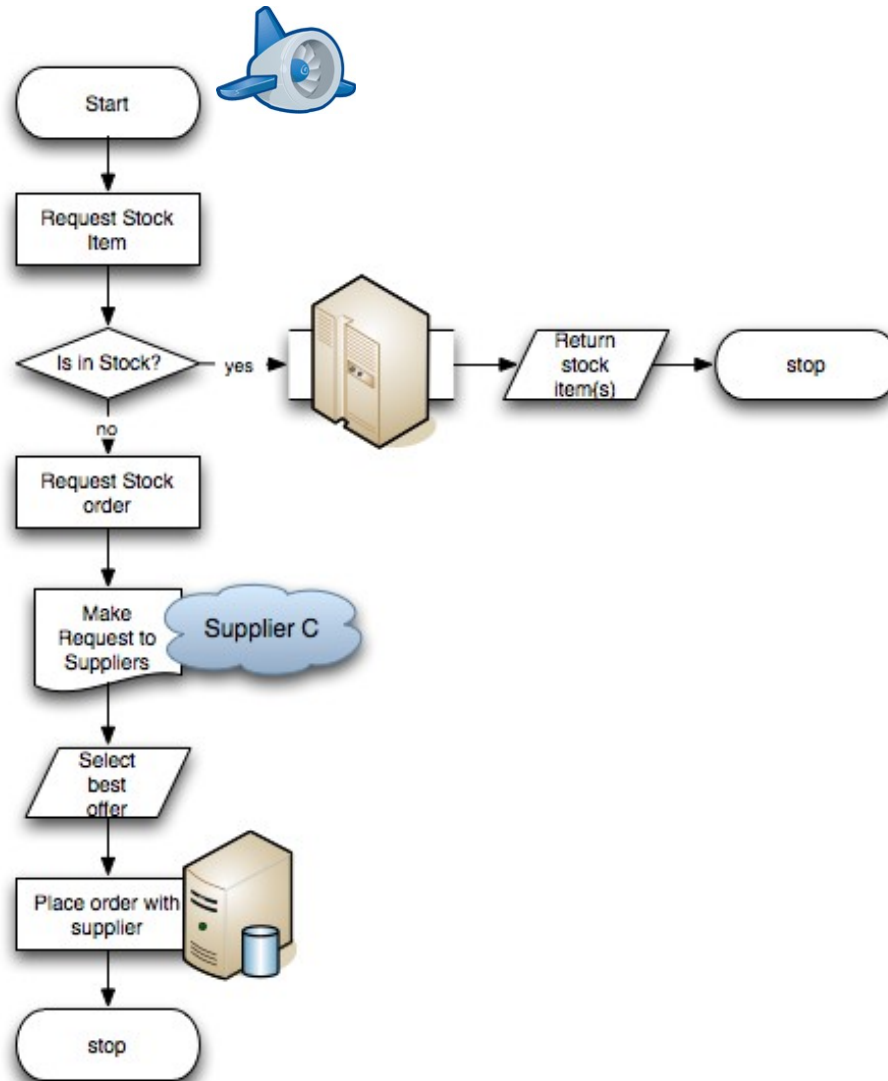


- ▶ Items are Requested
- ▶ If Items are in stock
  - Update Inventory
  - Return requested Items
- ▶ Otherwise
  - Create a Supplier Bid
  - Send to Suppliers
  - Select best bid offer
  - Place order with supplier

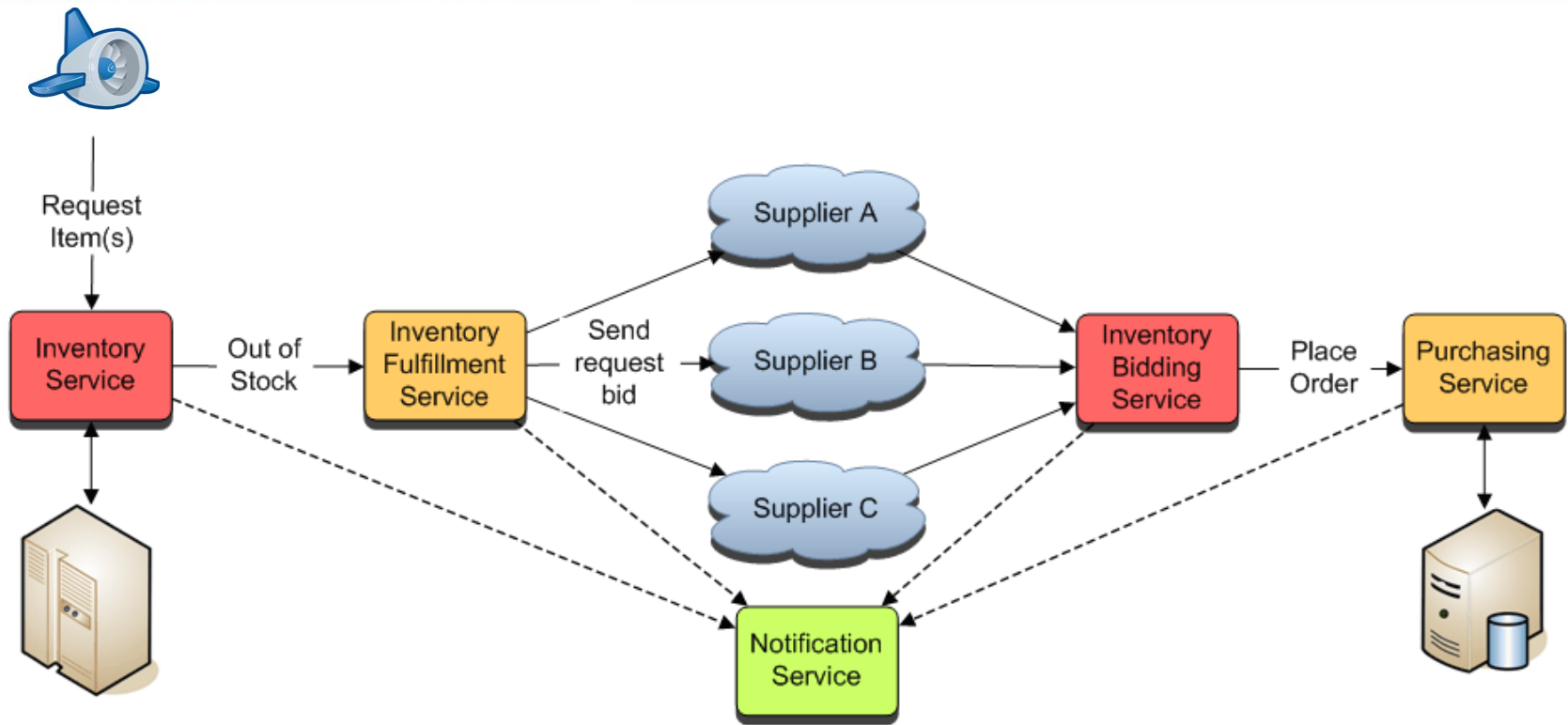


System	Details	Data in/out
	New Webstore: Google AppEngine	[in/out] HTTP (Rest)
	Inventory: Minicomputer system. Uses IBM AS-400 data queues to communicate.	[in/out] Fixed flat file.
	Purchasing: In house application, uses MQ Series as its communication interface.	[in] Xml – purchase.xsd
	Suppliers: We will assume that all our suppliers use web services. Secure Http should be used.	[in/out] Soap (mixed)
	We want to use a JMS message bus since we have tens of other systems to integrate later.	[in/out] bus Xml

# Interactions Between systems

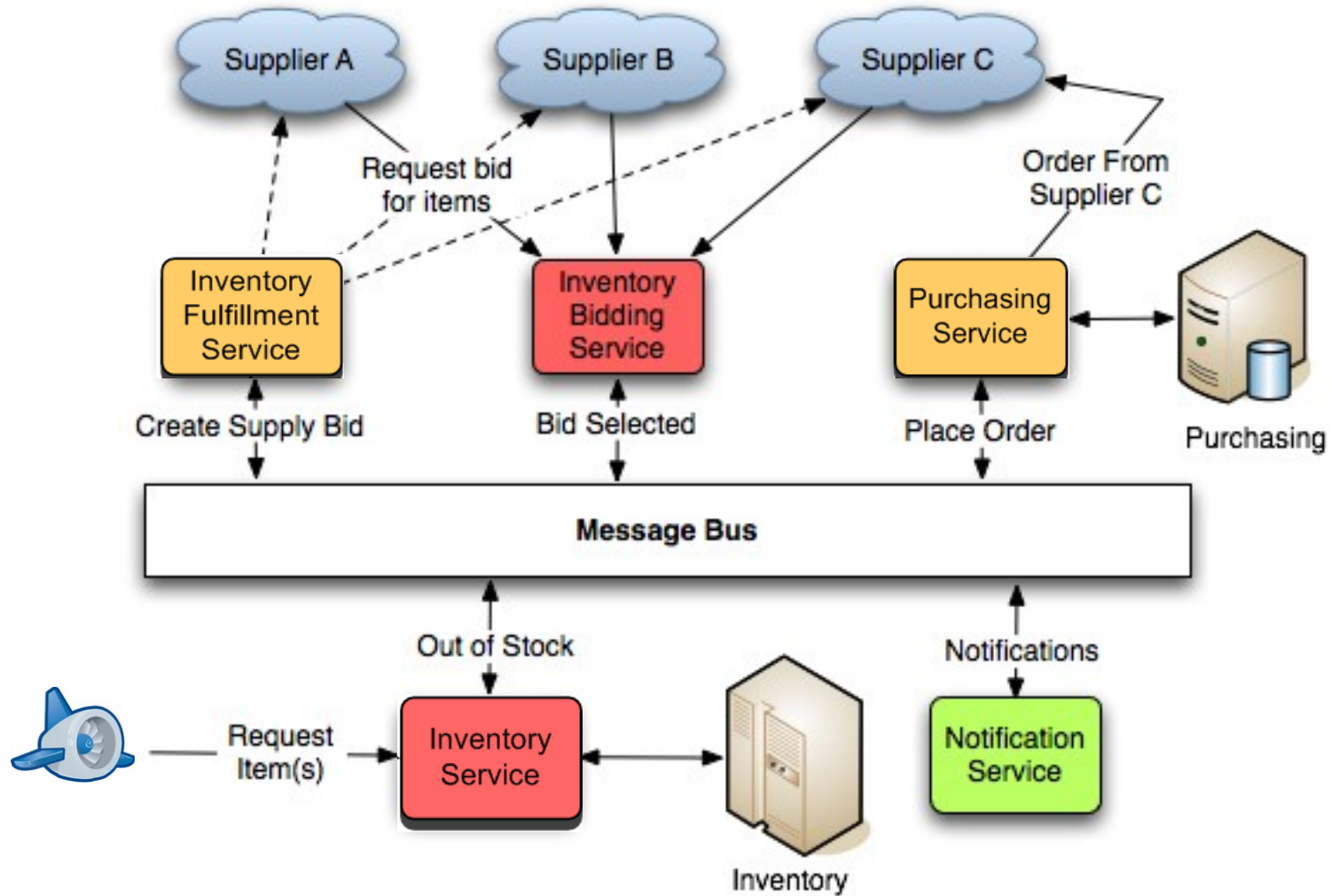


# Identify Services

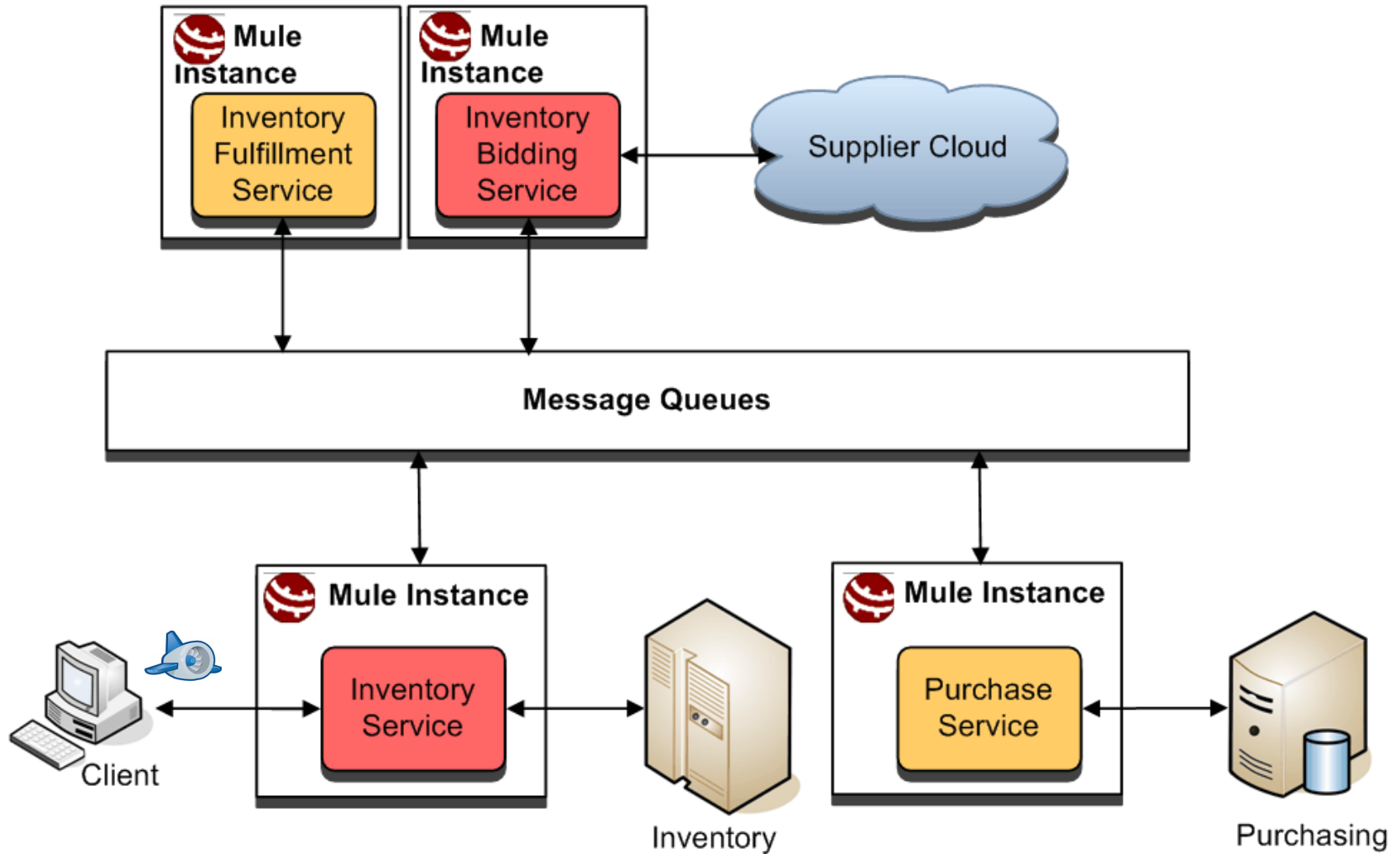


- ▶ Inventory Service – Interface to the inventory Application
- ▶ Inventory Fulfillment – Able to contact suppliers to bid for supply
- ▶ Inventory Bidding Service – receives bids and selects best supplier
- ▶ Purchasing Service – Interface to the inventory Application

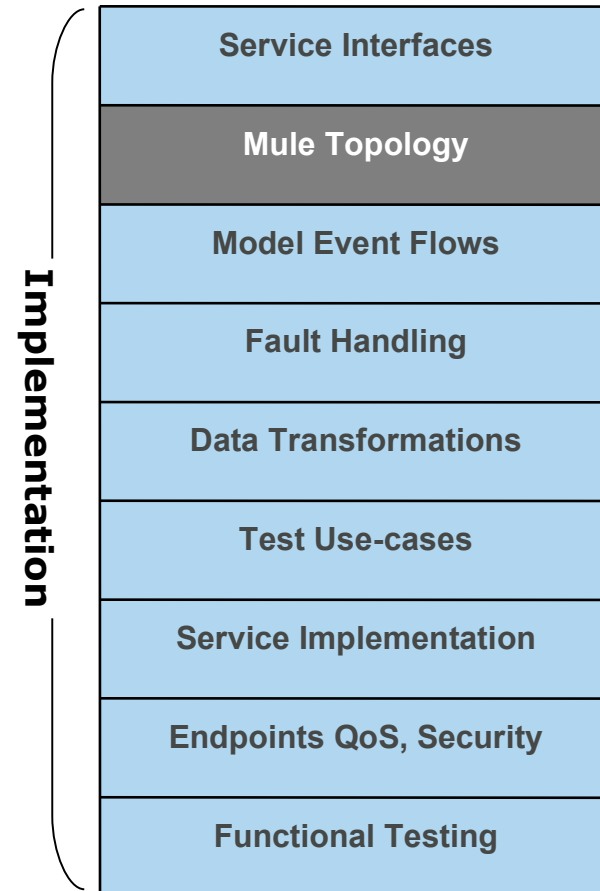
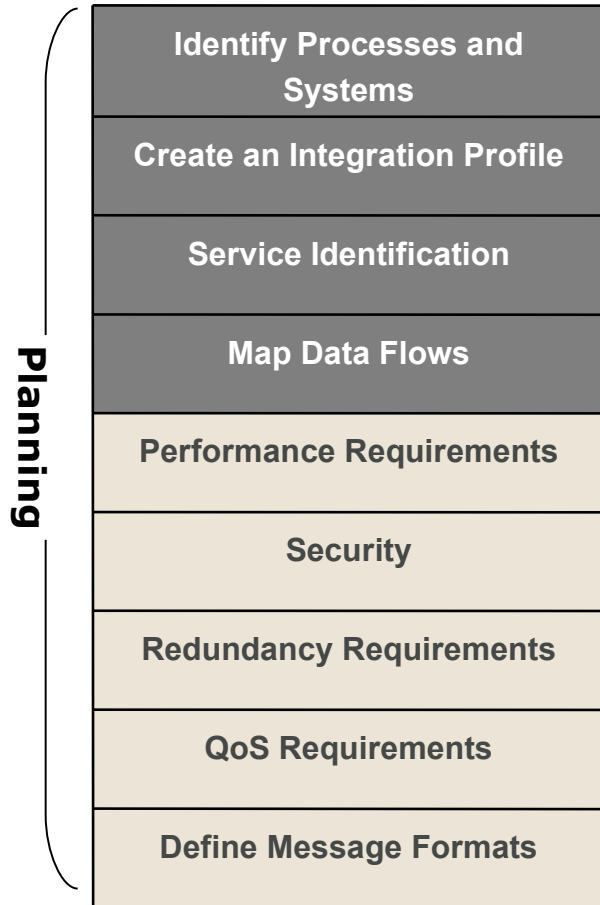
# Map Data Flows



# Mule Topology



# Checklist Recap



- ▶ Let's start with our Inventory Service



<b>Service Interfaces</b>
<b>Mule Topology</b>
<b>Model Event Flows</b>
<b>Fault Handling</b>
<b>Data Transformations</b>
<b>Test Use-cases</b>
<b>Service Implementation</b>
<b>Endpoints QoS, Security</b>
<b>Functional Testing</b>

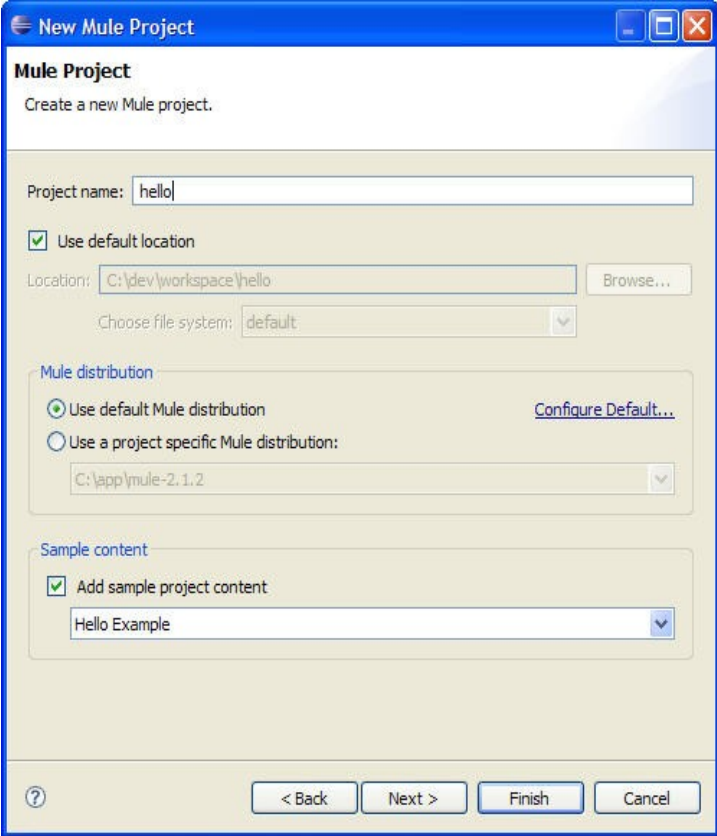
**Define the Service Interface**

**Create an Implementation**

**Create test case**

# Creating the project

- ▶ Introducing the Mule IDE
- <http://www.mulesource.org/display/MULEIDE/Home>



**New Mule Project**

**Mule Project**  
Create a new Mule project.

Project name:

Use default location

Location:

Choose file system:

**Mule distribution**

Use default Mule distribution [Configure Default...](#)

Use a project specific Mule distribution:

**Sample content**

Add sample project content

- ▶ Use the Mule project wizard to create a template project (using Maven 2.0.9) –

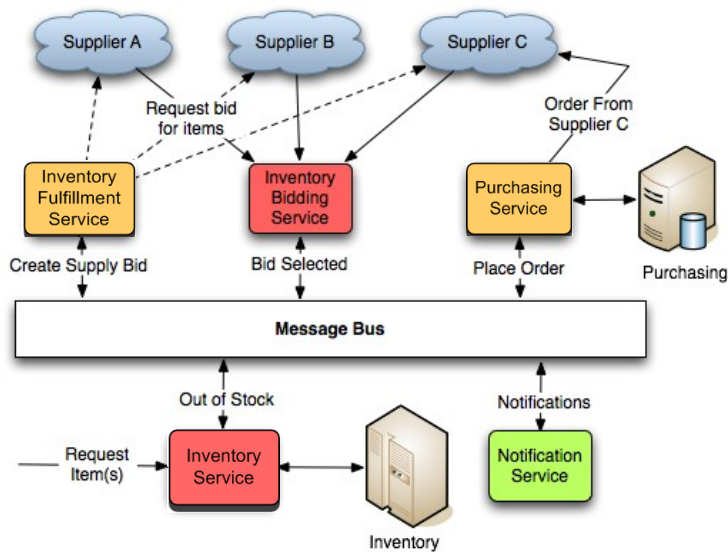
```
> mvn mule-project-archetype:create \  
    -DartifactId=inventory \  
    -DmuleVersion=2.1.2
```

- ▶ Then generate your project files –

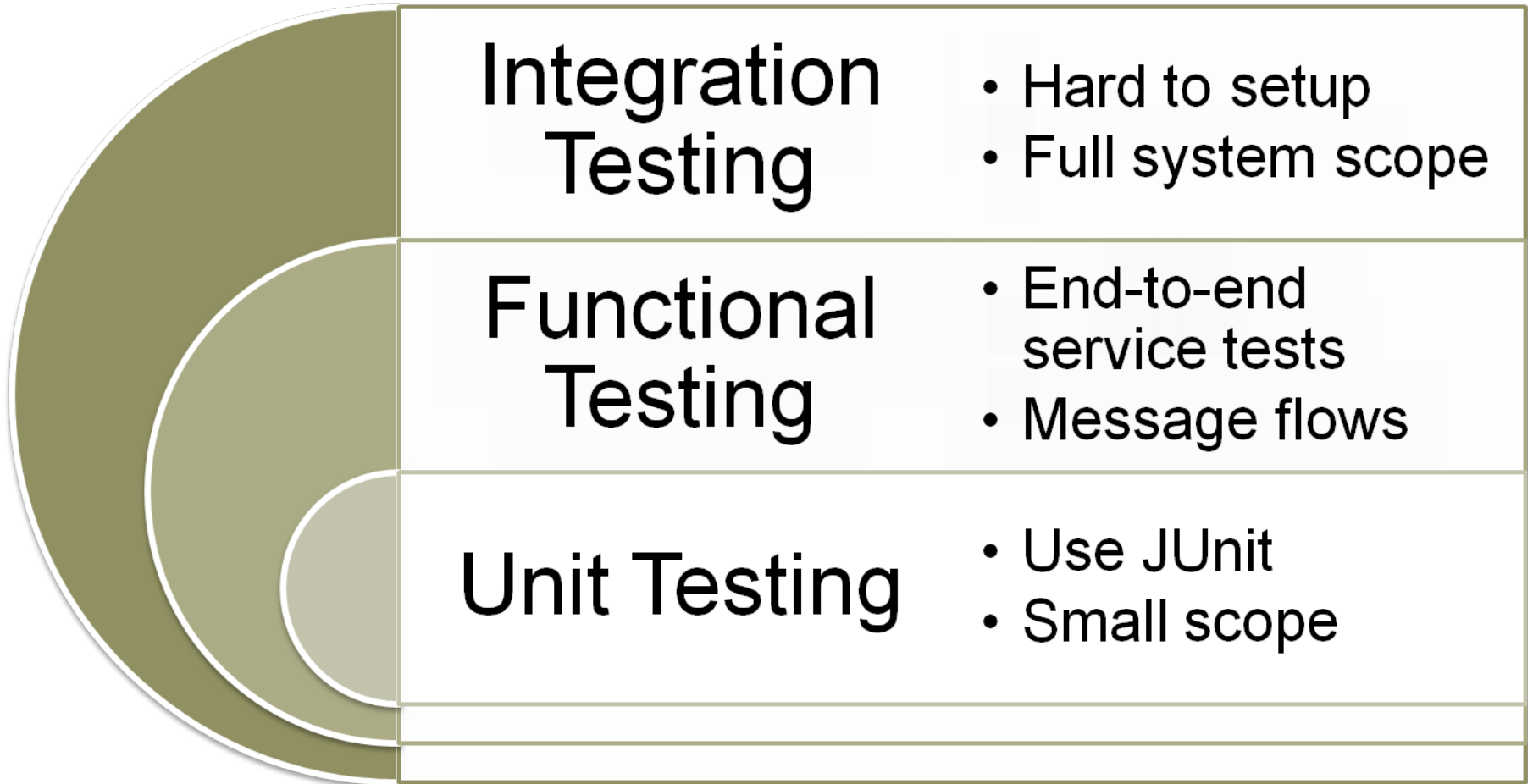
```
> cd inventory  
> mvn idea:idea  
  
> mvn eclipse:eclipse
```

- More info:

<http://www.mulesource.org/display/MULE2USER/Project+Archetype>



Service Interfaces
Mule Topology
Model Event Flows
Fault Handling
Data Transformations
Test Use-cases
Service Implementation
Endpoints QoS, Security
Functional Testing



**Create a skeleton Mule config**



**Wire “mock” services together**



**Use local endpoints for testing**



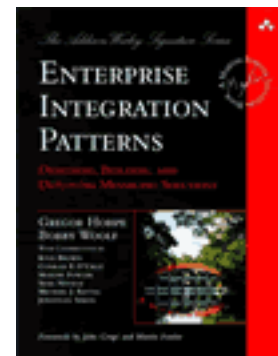
**Create test case**

- ▶ Mock out the Inventory Application

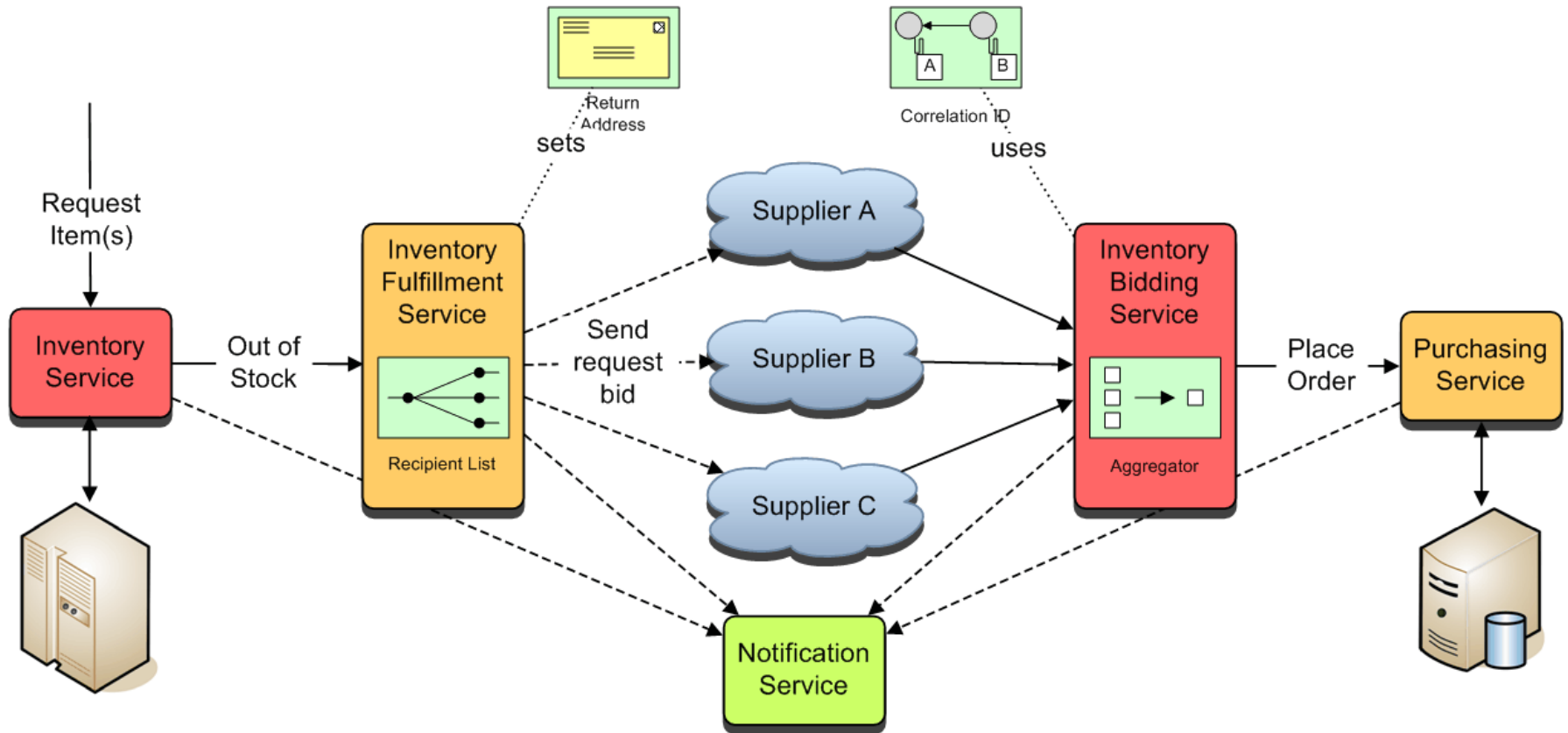
```
<service name="InventoryService">
  <inbound>
    <vm:inbound-endpoint path="TestInventoryService"/>
  </inbound>
  <test:component appendString="rcv'd by InventoryService"/>
  <outbound>
    <pass-through-router>
      <vm:outbound-endpoint path="FulfillmentRequest"/>
    </pass-through-router>
  </outbound>
</service>
```

To the IDE

- ▶ Integration is difficult
- ▶ Break down the problem into recognizable parts
- ▶ Common vocabulary
- ▶ Patterns embody expert knowledge
  - Best Practice
- ▶ EIP Book is a good reference manual
  - <http://www.enterpriseintegrationpatterns.com>



# Enterprise Integration Patterns



- ▶ Mule's flexible architecture
- Check out published customer use cases and technical webinars at MuleSource Resource Center
  - H&R Block: ESB integration broker for 13,000 office locations
  - OpSource: ESB data integration for In-the-Cloud integration
  - LLS: ESB backbone for Team in Training Fundraising Portal
  - Make Web Applications Do More With Mule
  - Implementing Web Service Messaging Patterns With Mule
  - Using and Configuring Security With Mule (4/14)
  - ... more



[http://mulesource.com/resource\\_center/](http://mulesource.com/resource_center/)

- ▶ Mule is flexible and customizable, has diverse usage
- ▶ Consuming and offering services in the Cloud is easy
- ▶ Deploy to and integrate with Cloud platforms
- ▶ Integration check list – best practices and guideline
- ▶ Creating a skeleton message flow – best practice
- ▶ Testing in Mule

# Now what?



Try Mule for yourself: <http://mulesource.com/download/>

Check out Mule blog: <http://blog.mulesource.org/>

Get training (leave your business cards to get a 20% discount):  
[training@mulesource.com](mailto:training@mulesource.com)

Read about Mule (Mule in Action, Open Source ESB in Action,  
Mule 2 (Apress))

Or, send me email at [ken.yagen@mulesource.com](mailto:ken.yagen@mulesource.com)