Mule - A Detailed Look at an Enterprise Service Bus

Tom Bender
Tendril Networks

tommybender@gmail.com
Introduction

- Adoption of SOA Architecture?
  - Commercial, Open Source, Rolled your own?

- Adoption of ESB Solution?
  - Commercial, Open Source, Rolled your own?

- What are your goals in attending this session?
  - What, Why and Where of ESB?
  - What makes Mule so cool?
  - Mule ESB Coding Samples?
Introduction

Tom Bender
Introduction
Introduction
Introduction
Introduction
Introduction
Introduction
Introduction
Introduction
Introduction

Systems Architect
Introduction

Industrial Machine Vision
Ann Arbor, Michigan
Introduction

Financial Quantitative Expert
Trading Systems
Los Angeles, CA
Introduction

Expert Constraint Systems
Sales Force Automation
Golden, CO
Introduction

Consultant
Boulder, CO
Introduction

Local Commerce

Jabber IM

Denver, CO
Introduction

Satellite Scheduling Systems
Geospatial Information Systems
Longmont, CO
Introduction

802.15.4
Wireless Sensor & Actuator Networks
Boulder, CO
Introduction

• Dave Chappelle?
Introduction

• Dave Chappelle

David Chappell

Enterprise Service Bus
Agenda

- Definition of Terms
- Introduction to Mule
- A Closer Look at Mule
- Inside Mule, A Look at SEDA
- Workflow Examples using Mule ESB
Technology Quote

"Perfection is achieved, not when there is nothing more to add, but when there is nothing left to take away."
- Antoine de Saint-Exupery
Definition of Terms

- **Service:**
  - is a coarse-grained, discoverable, and self-contained software entity that interacts with applications and other services through a loosely coupled, often asynchronous, message-based communication model.

- **Service Oriented Architecture:**
  - a collection of services with well-defined interfaces and a shared communications model is called a service-oriented architecture (SOA). A system or application is designed and implemented as a set of interactions among these services.

- **Enterprise Service Bus:**
  - provides a lightweight, loosely coupled, event-driven SOA with a highly distributed universe of named routing destinations across a multi-protocol message bus.
Definition of Terms

- **Workflow:**

  A workflow management system (WFMS) is a software component that takes as input a formal description of business processes and maintains the state of processes executions, thereby delegating activities amongst people and applications. - Tom Baeyens, jBPM Founder
Principles of SOA

“Four Tenets of SOA” as proposed by Don Box:

- Boundaries are Explicit
- Services are Autonomous
- Services share Schema and Contract, not Class
- Compatibility is based upon Policy
Properties of an ESB

- Light Weight
- Loosely Coupled
- Event-Driven
- Transactional
- Securable
- Distributed Network Topologies
- Abstract Endpoints
- Intelligent Routing
- Message Transformation (inbound/outbound)
- Reliable Messaging / QoS
- Multi-Protocol Message Bus
ESB Adoption: Partitioning and Encapsulating SOA

Integration Approaches

<table>
<thead>
<tr>
<th>Application and Integration Logic separated</th>
<th>Application and Integration logic not separated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traditional EAI</td>
<td>ESB</td>
</tr>
<tr>
<td>Application Servers</td>
<td>Custom Code/MOM</td>
</tr>
</tbody>
</table>

Hub-and-Spoke Integration

Distributed Integration
Introduction to Mule

- Brief Description
- Community
- Resources
Introduction to Mule

Description

- Mule is an event-based architecture.
- Actions within a Mule network are triggered by either events occurring in Mule or in external systems.
- Events always contain some sort of data, the payload.
- The payload is used and/or manipulated by components and a set of properties that are associated to the processing of the event.
- These properties are arbitrary and can be set at any time from when the event is created.
- The data in the event can be accessed in its original state or in its transformed state.
- The event will use the transformer associated with the Endpoint that received the event to transform its payload into a format that the receiving component understands.
Introduction to Mule

- Mule is a light-weight messaging framework.
- Highly distributable object broker
- Pluggable Connectivity for multiple transports and protocols
- Multiple Messages Exchange Types
- Web Services using Axis or Glue
- Multiple Topology Configurations
- Declarative Transactions including XA
- Staged Event Driven Architecture
- Dynamic, Declarative, content-based and rule-based Routing
- Message Transformation
- Event Routing based on Enterprise Integration Pattern
Introduction to Mule

Community

➢ Committers
  • Ross Mason, Founder
    ✓ ross.mason@symphonysoft.com
  • Guillaume Nodet
    ✓ gn@codehaus.org
  • Brian Topping
    ✓ topping@codehaus.org
  • Stéphane Vanmeерhaeghe

➢ Contributors
  • http://mule.codehaus.org/docs/team-list.html
  • http://mule.codehaus.org/Team
Introduction to Mule

- Resources
  - Web Site
  - Documentation
  - IRC
  - Source Code
  - Bug Database/JIRA
  - Mailing List Archive
  - Example Code
Introduction to Mule

- Web Sites
  - http://mule.codehaus.org
  - http://www.muleumo.org
Introduction to Mule

Documentation

- Mule Documentation Home
  - http://mule.codehaus.org/docs/
- Mule User Guide
  - http://docs.codehaus.org/display/MULE/User+Guide
- Mule Cookbook
  - http://docs.codehaus.org/display/MULE/Mule+Cookbook
- Mule Client
  - http://docs.codehaus.org/display/MULE/Mule+Client
- Mule Message Routers
  - http://mule.codehaus.org/Message+Routers
- Mule Properties Configuration
  - http://mule.codehaus.org/Configuring+Properties
Introduction to Mule

- **Mule Docs** *(Continued)*
  - JBoss Integration
    - [http://mule.codehaus.org/JBoss+Integration](http://mule.codehaus.org/JBoss+Integration)
  - Spring as a Component Factory
    - [http://mule.codehaus.org/Using+Spring+as+a+Component+Factory](http://mule.codehaus.org/Using+Spring+as+a+Component+Factory)

- **RSS Feeds**
  - Site
    - [http://docs.codehaus.org/spaces/rss.action?key=MULE&newPages=false](http://docs.codehaus.org/spaces/rss.action?key=MULE&newPages=false)
  - News
    - [http://docs.codehaus.org/spaces/blogrss.action?key=MULE](http://docs.codehaus.org/spaces/blogrss.action?key=MULE)
Introduction to Mule

- IRC
  - Information on IRC Options
    - http://mule.codehaus.org/IRC
  - IRC Channel
    - irc://irc.codehaus.org/#mule
Introduction to Mule

- Source Code
  - Mule Releases
    - [http://mule.codehaus.org/Download](http://mule.codehaus.org/Download)
  - CVS
    - [http://mule.codehaus.org/CVS+Info](http://mule.codehaus.org/CVS+Info)
Introduction to Mule

- Bug Database
  - JIRA
    - http://jira.muleumo.org/
Introduction to Mule

codehaus

- Mailing List
    - announce, despots, dev, scm, user

- GMANE Search
  - User
    - [http://search.gmane.org/?group=gmane.comp.java.mule.user](http://search.gmane.org/?group=gmane.comp.java.mule.user)
  - Dev
    - [http://search.gmane.org/?group=gmane.comp.java.mule.devel](http://search.gmane.org/?group=gmane.comp.java.mule.devel)
A Closer Look at Mule

- UMO Component
A Closer Look at Mule

- Wiring Applications with Mule

Diagram: Flowchart showing applications connected through a UMO Component.
A Closer Look at Mule

- Wiring Applications with Mule
A Closer Look at Mule

- Wiring Applications with Mule
A Closer Look at Mule

- Wiring Applications with Mule
A Closer Look at Mule

- Wiring Applications with Mule

Diagram:

- Application
- Channel
- Message Receiver
- Connector
- Transformer
- Inbound Router
- UMO Component
- Outbound Router
- Application
A Closer Look at Mule

Wiring Applications with Mule
A Closer Look at Mule

- Wiring Applications with Mule
A Closer Look at Mule

- **Channel**
  - Logical address in the messaging system

- **Mule Channel**
  - Wires components
  - Supports many channel options known as Transport Providers

---

EIP - Channel (60) http://www.eaipatterns.com/Channel.html

Mule doesn’t implement or mandate any particular channels though Mule does provide many channel options known as Transport providers.

“When an application has information to communicate, it doesn’t just fling the information into the messaging system, it adds the information to a particular Message Channel. An application receiving information doesn’t just pick it up at random from the messaging system; it retrieves the information from a particular Message Channel.”
A Closer Look at Mule

- **Message Receiver**
  - Receives data from the Application.
  - Supports Multiple Transport Providers
    - Jms, soap, http, tcp, xmpp, smtp, file, etc.
  - Additional Information

"Message Endpoint code is custom to both the application and the messaging system’s client API. The rest of the application knows little about message formats, messaging channels, or any of the other details of communicating with other applications via messaging. It just knows that it has a request or piece of data to send to another application, or is expecting those from another application. It is the messaging endpoint code that takes that command or data, makes it into a message, and sends it on a particular messaging channel. It is the endpoint that receives a message, extracts the contents, and gives them to the application in a meaningful way."
A Closer Look at Mule

- Connector
  - Sends and receives data over a channel
  - A message receiver is coupled with a connector and Transformer comprising a *Transport Provider*
  - The connector is responsible for sending data to a resource and managing a listener on the connector to receive data from a resource. (*e.g.* JmsConnector)
A Closer Look at Mule

- Connector
  - MessageReceiver
    - Used to listen on an endpoint to the underlying protocol.
  - MessageDispatcher
    - Is used to dispatch events to the underlying protocol.
    - Message Dispatchers are poolable threads that can be used to dispatch events or perform a receive on the underlying protocol to retrieve an available event if there is one.
    - The UMOMessageDispatcher interface defines 3 important methods:
      - dispatch() - sends data to the external system asynchronously.
      - send() - sends data to the external system synchronously and returns any response from the external system as a UMOEvent.
      - receive() - will request an event from the underlying technology and returns the result. This method has a timeout parameter.
A Closer Look at Mule

- Connector
  - Additional Information
A Closer Look at Mule

Transformer

- Transformers are used to convert source data to an object type required by the UMO Component.
- Transformers can be configured on Endpoints that receive data to ensure that the expected object type is always received by an UMO Component.
- Transformers configured on an Outbound endpoint ensure that the endpoint receives the the correct object type before dispatching the event.
- Multiple transformers can be chained together to allow for finer grained transformer implementations that are easier to reuse.
- To configure an Endpoint to use more than one transformer, just specify a space separated list of transformers in the config file.
A Closer Look at Mule

Transformer

- Inbound Transformer
  - JMS Text Message To String
  - Encrypted String To String
  - XML to Bean

- Outbound Transformer
  - String to Email Message
  - XSLT Transform To HTML
  - Bean To XML
A Closer Look at Mule

Endpoint

- a logical, configuration entity that is attached to components or external network resources and used to control *where* events are sent and received in Mule
  - **Endpoint URI** - an address that is used to reference a resource or service either locally or remotely. This must be a valid URI.
  - **Connector** - used to connect to the underlying transport. Often the connector is not explicitly set, instead the connector for the endpoint is found based on the scheme of the Endpoint URI.
  - **Filter** - a filter to apply to messages being received on the endpoint. See the Transports Guide for information about filtering support for a specific transport.
  - **Transaction** - Transactions can begin or commit when an event is received or sent.
  - **Properties** - These can be set to override certain properties on the connector for this endpoint instance. For example, when using a smtp endpoint you may want to overload the from address.
A Closer Look at Mule

- **Endpoint Configuration**
  - **POP3/SMTP**
    - pop3://user:password@mail.mycompany.com
    - smtp://user:password@mail.mycompany.com
  - **JMS Topic or Queue**
    - jms://topic:myTopic
  - **HTTP**
    - http://mycompany.com/mule
  - **File**
    - file:///tmp/data/in
  - **VM**
    - vm://MyUMO
  - **SOAP**
    - axis:http://mycompany.com/mule/services/MyUMO
A Closer Look at Mule

- Endpoint Configuration
  - Additional Information
    - http://mule.codehaus.org/Configuring+Endpoints

**Endpoint**
An endpoint is really a configuration wrapper that binds a connector, endpoint URI, transformers, filters and transactional information to provide a Channel Adapter. The provider also stores transactional information for the provider instance. See the Mule Endpoints chapter that goes into more detail about endpoints.

**EIP - Channel Adapter (127) [http://www.eaipatterns.com/ChannelAdapter.html](http://www.eaipatterns.com/ChannelAdapter.html)**

A provider is equivalent to a Channel Adapter.

"The adapter acts as a messaging client to the messaging system and invokes application functions via an application-supplied interface. Likewise, the Channel Adapter can listen to Application-internal events and invoke the messaging system in response to these events."
A Closer Look at Mule

- **Elements of a Transport Provider**
  - Connector, is responsible for connecting to the underlying resource
  - Message Receiver, is used to receive events from the system
  - Connector Dispatchers, are responsible for passing data to the system
  - Transformers are Transport specific transformers used to convert inbound and outbound data
A Closer Look at Mule

- **Message Routers**
  - Controls *how* events are sent and received by components in the system
  - Inbound Routers that are invoked when an event is received
  - Outbound Routers that are invoked when an event is being dispatched
  - Response Routers that are invoked when a registered event is received

- **Router Types**
  - Inbound
  - Outbound
  - Response
A Closer Look at Mule

**Inbound Routers**
- Idempotent Receiver (528)
- Selective Consumer (515)
- Aggregator (268)
- Resequencer (283)
- Forwarding Consumer

**Outbound Routers**
- Filtering Outbound Router (70)
- Recipient List (249)
- Multicast Router
- Chaining Router
- Message Splitter (259)
- Filtering List Message Splitter (70), (259)
A Closer Look at Mule

- Response Routers
  - Response Aggregator
    - Aggregation of responses from parallel requests prior to sending response to caller
A Closer Look at Mule

- Inbound Router
  - Controls *how* and *which* events are received by the attached UMO component
  - Filter, aggregate and re-sequence events
A Closer Look at Mule

- Outbound Router
  - Controls *how* and *which* components receive the dispatched event.
A Closer Look at Mule

- **Message Routers**
  - **Catch All Strategies**
    - No available routing path
    - Associate provider and route to common endpoint
  - **Additional Information**
    - Mule Document: [http://mule.codehaus.org/Message+Routers](http://mule.codehaus.org/Message+Routers)
A Closer Look at Mule

Filters

Filters provide the logic by which to invoke a particular router.
- Multiple Filter Types

Filters can be combined using logic filters.
- AndFilter
- OrFilter and
- NotFilter.
A Closer Look at Mule

- Filter Types
  - Payload Type Filter
  - RegEx Filter
  - Wildcard Filter
  - XPath Filter
  - Logic Filters
  - Message Property Filter

- Extensibility
  - Custom Filters

- Additional Information
  - [http://mule.codehaus.org/Filters](http://mule.codehaus.org/Filters)
A Closer Look at Mule

- Interceptors
A Closer Look at Mule

- Configuration
  - Mule DTD Reference
    - http://mule.codehaus.org/docs/mule-configuration.dtd.html
  - Mule DTD
A Closer Look at Mule

- Configuration
  - http://mule.codehaus.org/Configuration+Options
A Closer Look at Mule

```xml
<?xml version="1.0" encoding="UTF-8"?>


<mule-configuration id="fileProviderSampleProperties" version="1.0">
  <connector name="FileConnector"
    className="org.mule.providers.file.FileConnector">
    <properties>
      <property name="pollingFrequency" value="1000"/>
      <property name="outputPattern" value="${DATE}.xml"/>
      <property name="binaryMode" value="false"/>
    </properties>
  </connector>

  <model name="fileProviderSample">
    <mule-descriptor name="ProcessUMO"
      inboundEndpoint="file://inbound/myFiles"
      outboundEndpoint="file://outbound/myFiles"
      implementation="org.mule.samples.fileProvider.Component">
    </mule-descriptor>
  </model>
</mule-configuration>
```
A Closer Look at Mule

Configuration

Pooling Profile

• Pooling behavior of the Mule UMO Components
• Scoped to …

✓ <mule-model/>
  ❖ All Components
✓ <mule-descriptor/> contained in the Model
  ❖ Specific Component
## Pooling Profiles

A pooling profile is used to configure the pooling behaviour of UMO components. Each descriptor can set its own pooling profile or a default one can be set on the mule-configuration. The `org.mule.config.PoolingProfile` contains all the necessary values to create a pool of UMOComponent proxies.

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxActive</td>
<td>Controls the maximum number of Mule UMOs that can be borrowed from a Session at one time. When non-positive, there is no limit to the number of components that may be active at one time.</td>
<td>5</td>
</tr>
<tr>
<td>maxIdle</td>
<td>Controls the maximum number of Mule UMOs that can sit idle in the pool at any time. When non-positive, there is no limit to the number of Mule UMOs that may be idle at one time.</td>
<td>5</td>
</tr>
<tr>
<td>initialisationPolicy</td>
<td>Specifies the behaviour of the Mule UMO pool when the pool is exhausted:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>* POOL_INIT_NO_COMPONENT (0) : Will not load any components in the pool on startup</td>
<td></td>
</tr>
<tr>
<td></td>
<td>* POOL_INIT_ONE_COMPONENT (1) : Will load only the first component in the pool on startup</td>
<td></td>
</tr>
<tr>
<td></td>
<td>* POOL_INIT_ALL_COMPONENTS (2) : Will load all components in the pool on startup</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Determines how components in a pool should be initialised. The possible values are -</td>
<td></td>
</tr>
<tr>
<td>exhaustionAction</td>
<td>* WHEN_EXHAUSTED_FAIL (0) : will throw a NoSuchElementException</td>
<td></td>
</tr>
<tr>
<td></td>
<td>* WHEN_EXHAUSTED_BLOCK (1) : will block (invoke Object.wait(long) until a new or idle object is available.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>* WHEN_EXHAUSTED_GROW (2) : will create a new Mule and return it (essentially making maxActive meaningless.)</td>
<td></td>
</tr>
<tr>
<td>maxWait</td>
<td>Specifies the number of milliseconds to wait for a pooled component to become available when the pool is exhausted and the exhaustedAction is set to WHEN_EXHAUSTED_BLOCK.</td>
<td>4000</td>
</tr>
<tr>
<td>factory</td>
<td>A fully qualified classname of the pool factory to use with this pool profile. Implementations must implement org.mule.util.ObjectFactory</td>
<td></td>
</tr>
</tbody>
</table>
A Closer Look at Mule

Configuration

Threading Profiles

- Controls thread pool behavior in Mule
  - org.mule.config.ThreadingProfile
- Pool Executor
  - Doug Lea’s Concurrency APIs / JDK 1.5
# Threading Profiles

Threading Profiles are used to control how thread pools behave in Mule. Thread pools in Mule use Doug Lea's concurrency API so all the variables defined on `org.mule.config.ThreadingProfile` are synonymous with `PoolExecutor`.

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Id</code></td>
<td>The threading profile system Id is used to identify which system component should use the profile. This attribute is only used when declaring profiles on the mule-environment-properties element and on connector components where 'messageReceiver', 'messageDispatcher', 'component' or 'default' can be used. For all other cases it can be set to 'default'.</td>
<td>default</td>
</tr>
<tr>
<td><code>maxThreadsActive</code></td>
<td>Controls the maximum number of threads that can be executed at any one time in a thread pool.</td>
<td>10</td>
</tr>
<tr>
<td><code>maxThreadsIdle</code></td>
<td>Controls the maximum number of threads that can be inactive or idle in a thread pool before they are destroyed.</td>
<td>10</td>
</tr>
<tr>
<td><code>threadTTL</code></td>
<td>Determines how long an inactive thread is kept in the pool before being discarded. If the maximum pool size or queue size is bounded, then it is possible for incoming execute requests to block. There are five supported policies for handling this situation. Exception handling policy helps guard against lockup.</td>
<td>60000</td>
</tr>
<tr>
<td><code>poolExhaustedAction</code></td>
<td>WHEN_EXHAUSTED_RUN : (4) - The thread making the execute request runs the task itself. This policy helps guard against lockup. WHEN_EXHAUSTED_WAIT : (0) - Wait until a thread becomes available. This policy should, in general, not be used if the minimum number of available threads is zero, in which case a thread may never become available. WHEN_EXHAUSTED_ABORT : (3) - Throw a RuntimeException WHEN_EXHAUSTED_DISCARD : (1) - Throw away the current request and return. WHEN_EXHAUSTED_DISCARD_OKDEST : (2) - Throw away the oldest request and return.</td>
<td>WHEN_EXHAUSTED_RUN</td>
</tr>
<tr>
<td><code>maxBufferSize</code></td>
<td>Determines how many requests if any are queued when the pool is at maximum usage capacity. The buffer is used as an overflow.</td>
<td>0</td>
</tr>
</tbody>
</table>
A Closer Look at Mule

Configuration

Queue Profiles
A Queue Profile is used to describe the properties of an internal Mule queue. Internal queues are used to queue events for each component managed by Mule.

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxOutstandingMessage</td>
<td>Defines the maximum number of message that can be queued.</td>
<td>1000</td>
</tr>
<tr>
<td>persistenceStrategy</td>
<td>A persistence strategy defines the mechanism used to store Mule events to a persistent store. Primarily, this is used for persisting queued events to disk so that the server's internal state is mirrored on disk in case the server fails and needs to be restarted. If no strategy is set the any queues created from this profile are not persistent.</td>
<td>null</td>
</tr>
</tbody>
</table>
A Closer Look at Mule

### Configuring Programmatically

Developers can use the `QuickConfigurationBuilder` to programmatically build a Mule Manager instance. This can be especially useful when configuring a Mule Instance from script or for test cases. You can configure a server with only a few lines of code, for example to configure a server with a single component:

```java
QuickConfigurationBuilder builder = new QuickConfigurationBuilder();
builder.createStartedManager(true, "tcp://localhost:60504");

UM0Endpoint inboundEndpoint = new MuleEndpoint("axis:http://localhost:81/services");
builder.registerComponent(EchoComponent.class.getName(), "echoComponent", inboundEndpoint);
```

This creates a new Mule Instance with a server admin url of 'tcp://localhost:60504' (if you don't want the Mule Admin Agent to start, set this to ''). I also creates a Mule Component called 'echoComponent' that is an Axis web service that will receive requests on 'http://localhost:81/services/echoComponent'.

The following example creates an asynchronous Mule Instance without an Admin Agent with a single component that receives events via tcp and writes them out to file.
A Closer Look at Mule

Alternate Configuration Builders

- Spring Integration
  - Using Spring as a Component Factory
  - Configuring the Mule Server From a Spring Context
  - Configuring a Spring context using Mule XML
  - Mule Events in Spring

- Additional Information on Spring Integration
  - [http://mule.codehaus.org/Spring+Integration](http://mule.codehaus.org/Spring+Integration)

- Inversion of Control / Dependency Injection Containers
  - Pico
  - Plexus
A Closer Look at Mule

- **Mule Manager**

![Mule Manager Diagram]

- **IoC Container**
- **Spring/Pico/Plexus**
- **Security Manager**
- **Transaction Manager**
- **Notification Manager**
- **Agents**
- **Jmx Agent**
- **Mule Admin Agent**
- **Discovery Agent**

**Mule Model (SEDA)**

- **Managed Components**
- **Lifecycle Adapter**
- **Endpoint Resolver**

**Mule Transport**

- **Xml Transformers**
- **Object Transformers**
- **Message Routers**
- **Filters**
- **Transport Providers**

- **Jms**
- **Soap**
- **Http**
- **Multicast**
- **Tcp**
- **Vm**
- **File**
- **Xtrace**
- **Xmpp**
- **Pop3**
- **Ssmtp**
A Closer Look at Mule

The Model

- Encapsulates and manages runtime behavior of a Mule Server Instance.
- The details describing this instance are contained in the Mule Configuration file.
A Closer Look at Mule

- Two Control Mechanisms to determine how Mule invokes a UMO component:
  - Entry Point Resolver
  - Life Cycle Adapter
A Closer Look at Mule

- Entry Point Resolver
  - org.mule.umo.model.UMOEntryPointResolver interface
  - Determines which method to invoke on an UMO component when event is received
  - Extend EntryPointResolver
A Closer Look at Mule

- Entry Point Resolver
  - Checks to see if the component implements the Callable lifecycle interface, then the onCall(UMOEvent) method will be used to receive the event.
  - If the component has a transformer configured for it, the return type for the transformer will be matched against methods on the component to see if there is a method that accepts the transformer return type. If so this event will be used. Note if there is more than one match, an exception will be thrown.
  - If there is a method on the component that accepts an org.mule.umo.UMOEvent. If so this event will be used. Note if there is more than one match, an exception will be thrown.
  - The last check determines if there are any methods on the component that accept a java.util.Event. If so this event will be used. Note if there is more than one match, an exception will be thrown.
  - If none of the above find a match an exception will be thrown and the component registration will fail.
A Closer Look at Mule

**Lifecycle Adapter**

- **Default Constructor**
  - Called when the component is resolved

- **Initialise**
  - Org.mule.umo.lifecycle.initialisable
  - Called when the component is registered with the Mule Model

- **Start**
  - Org.mule.umo.lifecycle.Startable
  - Called when the server is started

- **OnCall**
  - Org.mule.umo.lifecycle.Callable
  - Called when an event is received for the component. By configuring an EntryPointResolver this method can be resolved to any method on your object

- **Stop**
  - Org.mule.umo.lifecycle.Stoppable
  - Called when the Mule server is Stopped. The server can be Restarted.

- **Dispose**
  - Org.mule.umo.lifecycle.Disposable
  - Called when the Mule server is Shutdown. Clear up any resources.
A Closer Look at Mule

Events

- Asynchronous
  - Many events can be processed by the same component at the same time in different threads.

- Synchronous
  - Entire request processed in a single thread of execution.

- Request-Response
  - Request for an event, wait predefined timeout for response to enable aggregation of intermediary responses.
A Closer Look at Mule

- Asynchronous Events
A Closer Look at Mule

- Synchronous Events
A Closer Look at Mule

- **Transactions**
  - Agnostic to the underlying transaction manager
  - Mule transactions are configured on inbound endpoints

- **XA Transactions**
  - XA transactions can be used if you want to enlist multiple managed resources within the same transaction.
  - Three providers support XA transaction (i) VM (ii) JDBC (iii) JMS

- **JOTM**
  - Transaction support outside of application server.

- **Additional Information**
  - [http://mule.codehaus.org/Transaction+Management](http://mule.codehaus.org/Transaction+Management)
A Closer Look at Mule

- **Security**
  - authenticate requests *via* endpoints using transport specific or generic authentication methods.

- **Securing Mule**
  - Configure a Security Manager on the Mule Manager
    - Setup one or more Security Providers
      - Acegi Security
  - Configure the Security filter on the object to secure.

- **Additional Information**
  - [http://mule.codehaus.org/Mule+Security](http://mule.codehaus.org/Mule+Security)
A Closer Look at Mule

- JNDI
  - [http://mule.codehaus.org/Jndi+Container](http://mule.codehaus.org/Jndi+Container)
A Closer Look at Mule

- Mule Client
  - [http://mule.codehaus.org/Mule+Client](http://mule.codehaus.org/Mule+Client)
A Closer Look at Mule

Configuring Jms

This page describes the specifics for setting up various Jms Servers in Mule. For more information about all Mule Jms configuration go here. The following Jms server configurations are described -

- ActiveMQ
- JBoss MQ
- Joram
- OpenJms
- Oracle AQ
- SeeBeyond
- Spirit Wave
- UberMQ
- Weblogic Jms
- IBM WebSphere MQ

If you have configuration for a Jms server not listed here or there is a mistake on this page please raise a jira to get the document updated. Thanks.

✅ JMS Endpoint URIs and JNDI destinations

Some JNDI implementations treat dot (.) and forward slash symbols differently in destination names, so jms://order/incoming may not be the same as jms://order.incoming, but the former will not give you the order/incoming destination, but incoming. If you are dealing with such a server (JBoss is known to behave this way), here is a trick to help you:

```
jms://order/incoming
```

See Mule Endpoint URIs for reference. and JmsEndpointTestCases for some more examples.

⚠️ The following are just examples and configuration values will change depending on your application environment.
A Closer Look at Mule

- Configuring JMS
  - [http://mule.codehaus.org/Configuring+Jms](http://mule.codehaus.org/Configuring+Jms)
Configuring Properties

- Most of the configuration elements in the Mule XML configuration can have a `<properties>` element. This defines a set of bean properties that will be set on an object using standard getters and setters. As such, the properties names themselves must follow the JavaBean naming convention where the 'get' or 'set' part of the property method is removed and the first character of the property name is lowercase. So for a bean that has a methods getTestProperty() and setTestProperty(...) the property name is testProperty.

  ```xml
  <properties>
    <property name="testProperty" value="foo"/>
  </properties>
  ```

- If a property is of type boolean, int, long, double, byte or float it will be automatically converted to the correct type. Properties can also be Maps, Lists arrays, System properties, objects from a configured IoC container and factory objects. Each of these are described in more detail below.

- Addl Info: [http://mule.codehaus.org/Configuring+Properties](http://mule.codehaus.org/Configuring+Properties)
Inside Mule, A Look at SEDA

- Staged Event Driven Architecture
  - Matthew David Welsh
    - Harvard, Assistant Professor of Computer Science
    - University of California, Berkeley (1999)
    - NBIO: Nonblocking I/O for Java
      - java.nio package
      - SEDA & NBIO on Sourceforge
        - http://sourceforge.net/projects/seda
    - Additional Resources
      - http://www.eecs.harvard.edu/~mdw/proj/seda/
Inside Mule, A Look at SEDA

- Staged Event Driven Architecture Goals
  - Support massive concurrency
  - Simplify the construction of well-conditioned services:
  - Enable introspection:
  - Support self-tuning resource management:
Inside Mule, A Look at SEDA

Staged Event Driven Architecture

- decompose complex, event-driven applications into a set of stages connected by queues.
- avoids the high overhead associated with thread-based concurrency models, and decouples event and thread scheduling from application logic.
- By performing admission control on each event queue, the service can be well-conditioned to load, preventing resources from being overcommitted when demand exceeds service capacity.
- SEDA employs dynamic control to automatically tune runtime parameters (such as the scheduling parameters of each stage), as well as to manage load, for example, by performing adaptive load shedding.
Inside Mule, A Look at SEDA

- Staged Event Driven Architecture
  - An Architecture for Highly Concurrent, Well-Conditioned Internet Services
ESB Related Standards: Impacts and Considerations

- JSR 208 - Java Business Integration
- BPEL - Business Process Execution Language
- XPDL
- Contrast and Compare XPDL & BPEL
ESB Related Standards: Impacts and Considerations

- Java Business Integration
  - JBI provides the core standards required to build SOA-based integration server software. It focuses on standardising the interoperation semantics (and associated bindings and interfaces) between what it calls service engines and a normalized message router, which links the engines together.
ESB Related Standards: Impacts and Considerations

- **JBI Components**
  - **Service Engine**
    - The term Service Engine (SE) is used to refer to the portion of the JBI environment (or framework) that is responsible for exposing an application programming model to developers.
    - JBI does not define an application programming model but rather defines a set of SPIs that enable the development of standard Web Service “containers” that can seamlessly utilize multiple communication infrastructure ranging from HTTP/SOAP [ref.] to JMS/MOM [ref.] and including specialized AS1/AS2 EDI [ref.] communications stacks.
  - **Binding Component**
    - A Binding Component may choose to implement one or more communications protocols thus offering connectivity services to SEs and thereby enabling SEs to expose their services to local and remote consumers as well as enabling the consumption of remote and local Web Services.
ESB Related Standards: Impacts and Considerations

- ESB Related Standards
  - JMX & Web based Admin tools
  - ESB Life cycle & Services
  - Reconfiguration Deployment
    - Control
    - Monitoring
  - BC Life cycle & Services
  - Service Engine (SE) Framework
    - BPEL SE
    - Other SEs
  - Normalized Message Service
  - Binding Comp (BC) Framework
    - WS-I BC
    - Other BCs
  - JBI Environment
  - WSDL-based Service Providers & Consumers

Diagram:
- JMX & Web based Admin tools
- ESB Life cycle & Services
- Reconfiguration Deployment
  - Control
  - Monitoring
- BC Life cycle & Services
- Service Engine (SE) Framework
  - BPEL SE
  - Other SEs
  - Normalized Message Service
- Binding Comp (BC) Framework
  - WS-I BC
  - Other BCs
- JBI Environment
- WSDL-based Service Providers & Consumers
ESB Standards: Impacts and Considerations

- Mule-JBI Update
  - The Mule team are currently working on Mule-JBI
    - a JBI (JSR-208) implementation that reuses the core mule architecture
  - Alpha Release available September 2, 2005; Beta TBA
    - All Mule components, transports, routers and transformers can execute in any JBI container
  - Mule-JBI 1.0 will support the current alpha functionality, in addition to:
    - allow 3rd-party JBI components to be managed in the Mule container.
    - Existing Mule users will be able to integrate with JBI containers including Mule-JBI.
ESB Related Standards: Impacts and Considerations

- Workflow Management Coalition ("WfMC")
  - Founded in August, 1993
  - Non-profit, international organization of workflow stakeholders
  - Mission
    - Increase the value of customers’ investment with workflow technology
    - Decrease the risk of using workflow products
    - Expand the workflow market through increasing awareness for workflow
  - Standards
    - Wf-XML
    - XPDL - Process Definition Language
ESB Standards: Impacts and Considerations

Executional processes versus a WFMS:
State of Workflow - Tom Baeyens jBPM

- A recent trend in the BPM community is the convergence of specifications about executional business processes. The approach promoted by XLANG, WSFL and BPML converged into BPEL. BPEL is based on interactions (message exchanges). BPEL is defined in the context of a service-oriented architecture. One of the prerequisites is that the services to be addressed are described in a WSDL declaration. BPEL then specifies an XML grammar that can be seen as a programming language to combine control flow with calls to the services defined in WSDL.

ESB Standards: Impacts and Considerations

State Based Workflow vs Executional Business Process

- State based WFMSs are centered around the notion of state (or activity). The workflow engine maintains the state and calculates the transitions from one state to the next.

- On the other hand, executional business processes like BPEL are centered around the definition of reactions upon incoming message. A set of those reactions, along with some other bells and whistles, can be seen as a business process. That explains why BPEL is somewhat complementary to state based WFMSs. A BPEL onMessage event handler, which is a reaction upon an incoming message, could be executed e.g. on transitions between states.
Building Enterprise Applications with Mule

- **Workflow Enterprise Integration Patterns**
  - Itinerary-based Routing
    - Message Splitter (259)
    - Message Aggregator (268)
  - Content-based Routing
    - Composed Message Processor (294)
      - Message Splitter (259)
      - Content-based Router (230)
  - Context-based Routing
    - Dynamic Router (243)
      - Message Router (78)
Mule - A Detailed Look at an Enterprise Service Bus

Thank you
Building Enterprise Applications with Mule

- Loan Broker (445)
  - Translator (85)
  - Aggregator (268)
  - Recipient List (249)
References

- Mule Architecture Guide & Published Works, Symphony Soft
  - Ross Mason

- SEDA: An Architecture for Well-Conditioned, Scalable Internet Services

- Software Engineering Institute, Carnegie Mellon
  - http://www.sei.cmu.edu/

- Enterprise Service Bus
  - David A. Chappell - 2004

- Super-Peer Architectures for Distributed Computing
  - Fiorano - 2003

- The Evolution of Internet Scale Event Notification Services: Past, Present, and Future
  - Adam Rifkin and Rohit Khare

- JBI Specification - JSR 208
  - Sun Microsystems
References

- LogicBlaze, ActiveMQ, ServiceMix
  - James Strachen
- Constructing Software For Service Oriented Architecture, Attachmate
  - Jean-Jacques Dubray, Ph.D.
- JBI, The Only Game in Town
  - Ronan Bradley - 2005
- Microsoft Web Site
- Better, Faster, Lighter Java
  - Bruce Tate
- Workflow Management Coalition Specification
  - The Workflow Reference Model
  - http://www.wfmc.org/standards/docs.htm
- The State of Workflow
  - Tom Baeyens, jBPM Founder
References

- Fundamentals of Service Orientation, Attachmate
  - Jean-Jacques Dubray

- Enterprise Integration Patterns, Addison Wesley
  - Gregor Hohpe & Bobby Woolf