Service Oriented Integration with ServiceMix and Camel

Chris Custine
Principal Engineer
IONA Technologies

Agenda

- What is an ESB?
- What is JBI?
- Introduction to ServiceMix
- Introduction to Camel
- ServiceMix and Camel Working Together
- Roadmap for Servicemix 4.0

What is...

an ESB?

What is an ESB?

"An Enterprise Service Bus (ESB) is a new architecture that exploits Web services, messaging middleware, intelligent routing, and transformation. ESBs act as a lightweight, ubiquitous integration backbone through which software services and application components flow." (Gartner)

What is an ESB?

An ESB acts as a shared messaging layer for connecting applications and other services throughout an enterprise computing infrastructure. It supplements its core asynchronous messaging backbone with intelligent transformation and routing to ensure messages are passed reliably. Services participate in the ESB using either web services messaging standards or JMS (LooselyCoupled.com)

What is an ESB?

An ESB is an open standards, message-based, distributed, integration solution that provides routing, invocation, and mediation services to facilitate the interactions of disparate distributed information technology resources (applications, services, information, platforms) in a reliable manner.

(Brenda Michelson, Elemental Links)

What is...

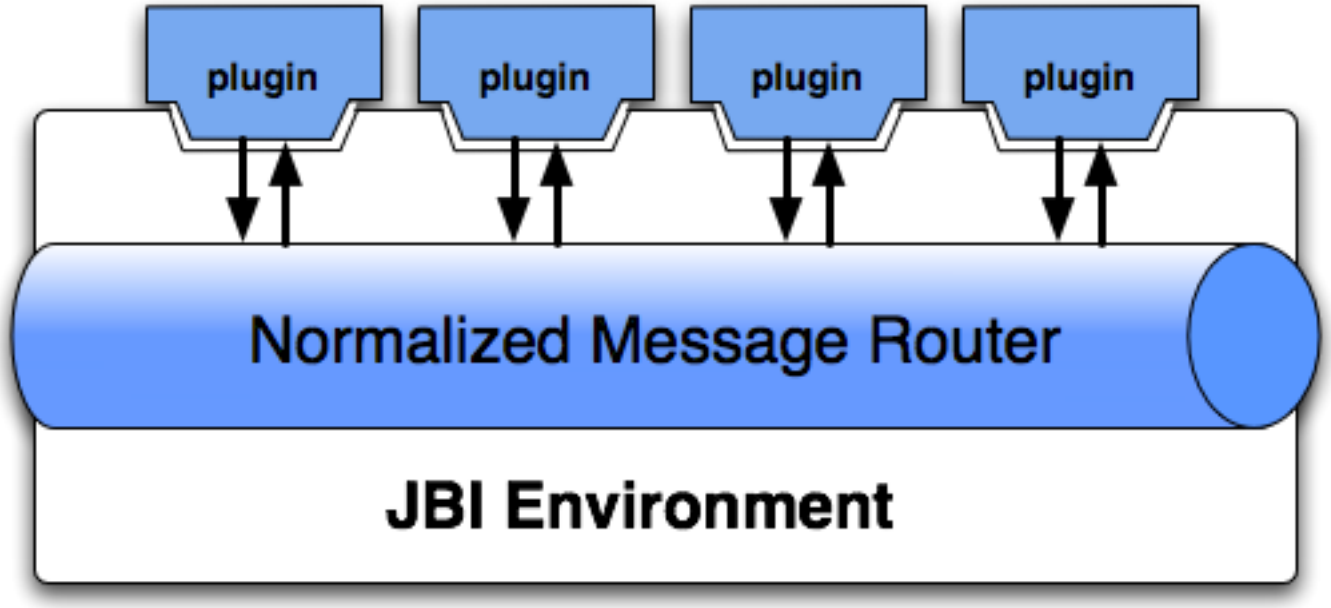
JBI?

What is JBI?

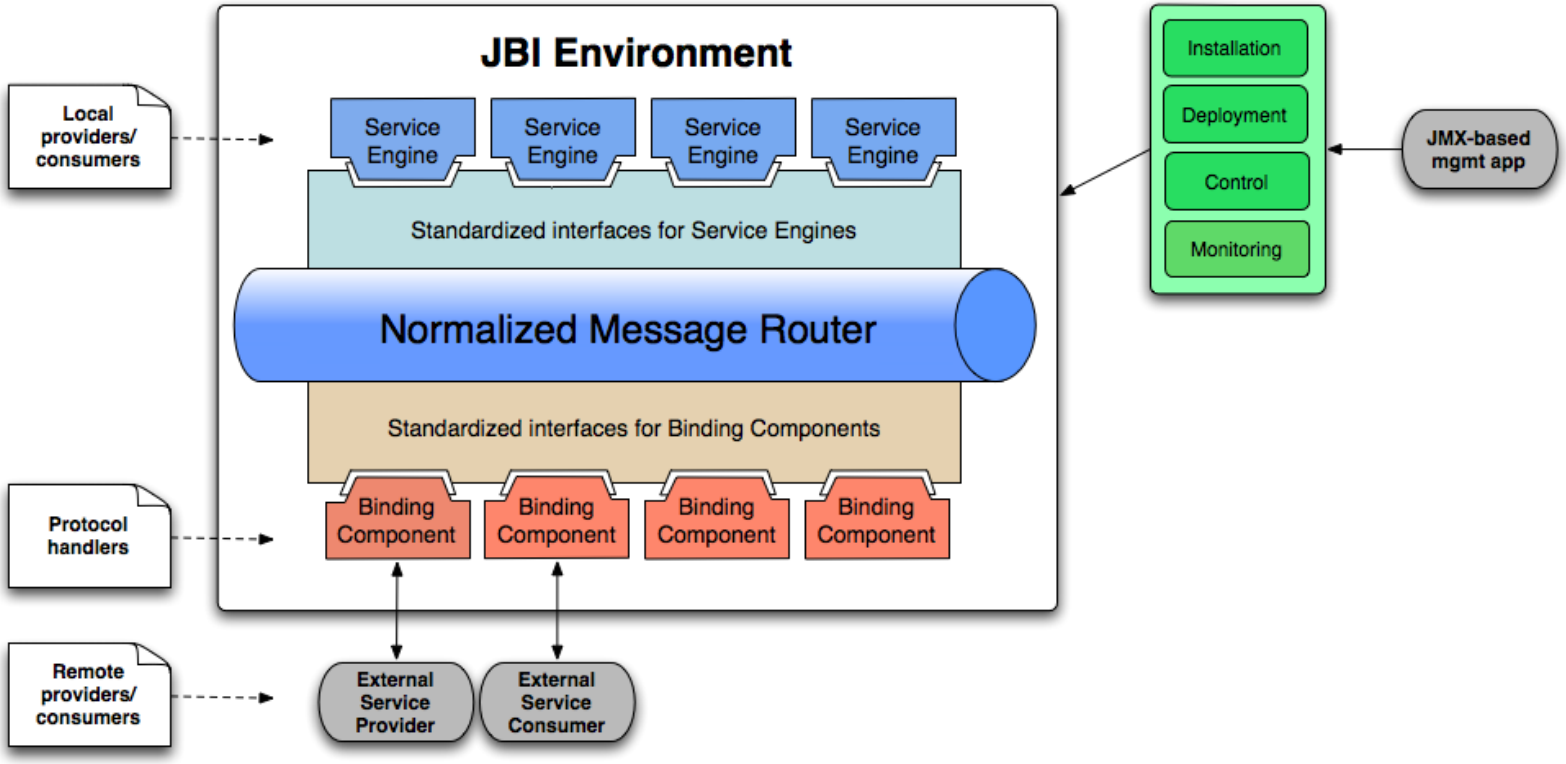
JBI defines an architecture that allows the construction of integration systems from plug-in components, that interoperate through the method of mediated message exchange.

(JBI 1.0 Spec)

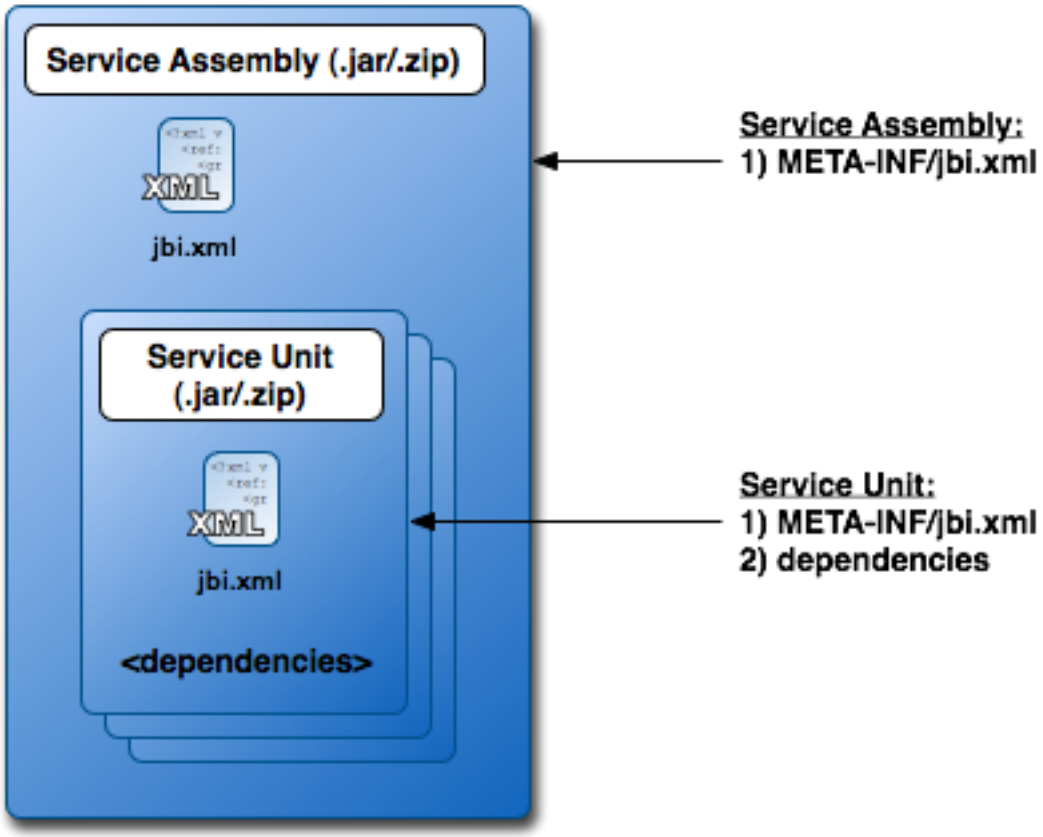
Java Business Integration



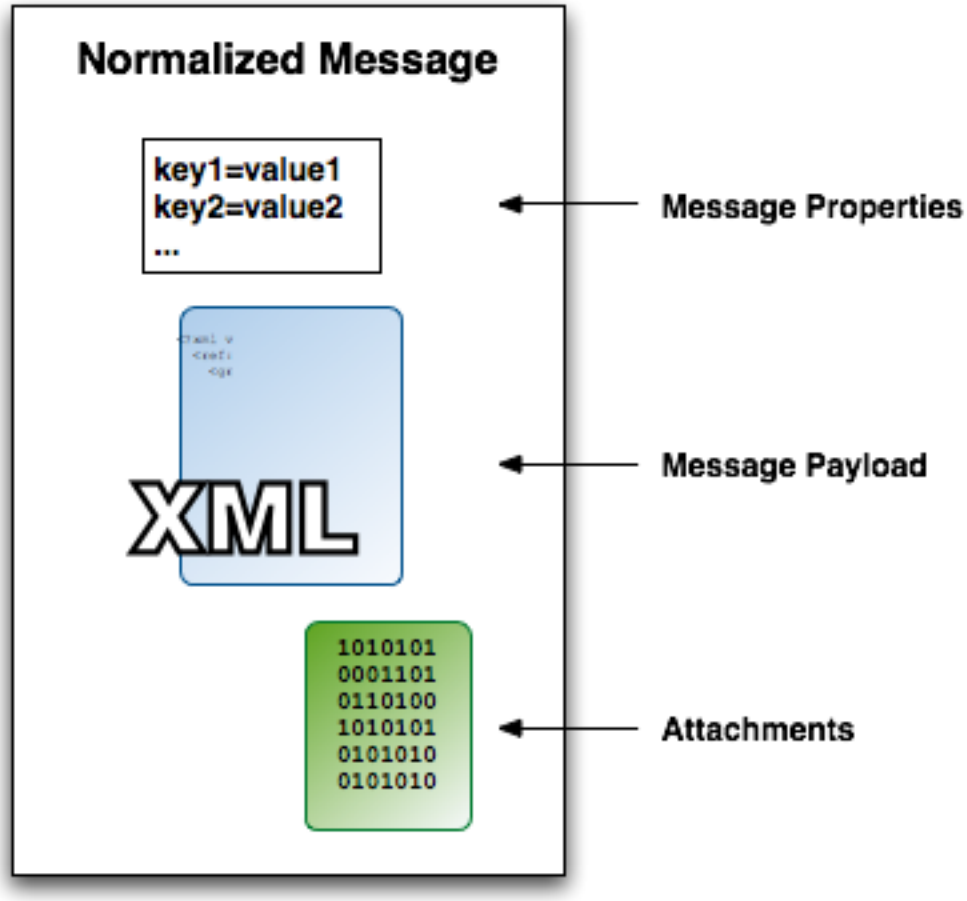
Java Business Integration



JBI Packaging



JBI Normalized Message

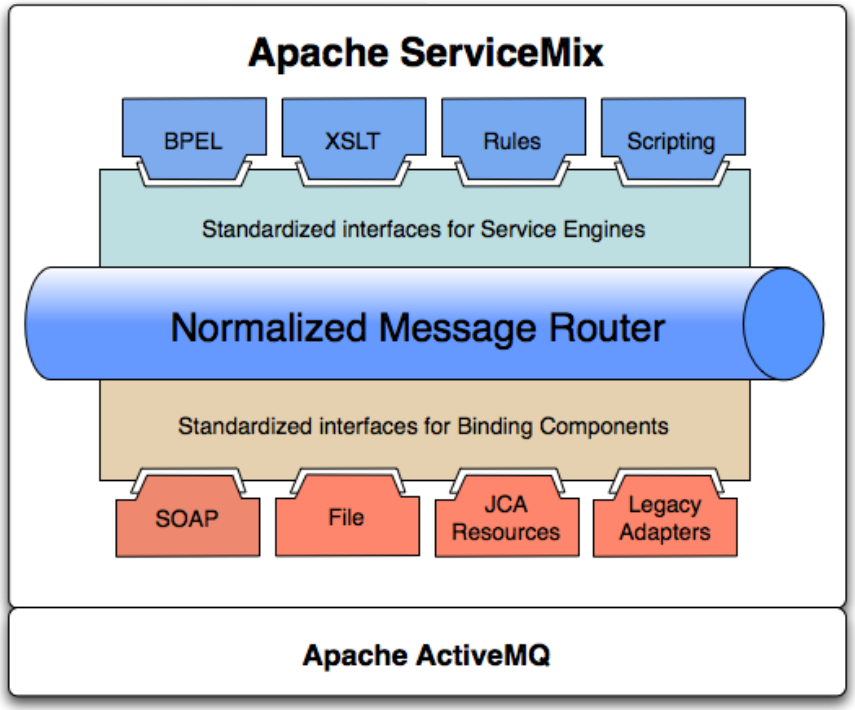


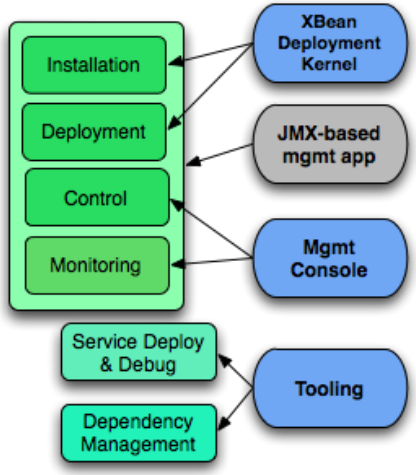
Introduction to ServiceMix

ServiceMix

The Apache Software Foundation
http://www.apache.org/

Apache ServiceMix Architecture





ServiceMix Container Features

- Runtime Container
  - Package Deployment
  - Lifecycle Management
  - Monitoring and Metrics
- Automatic message mediation
  - STP and SEDA Flows for high performance local routing
  - JMS and JCA Flows for distributed routing
  - Utilizes ActiveMQ

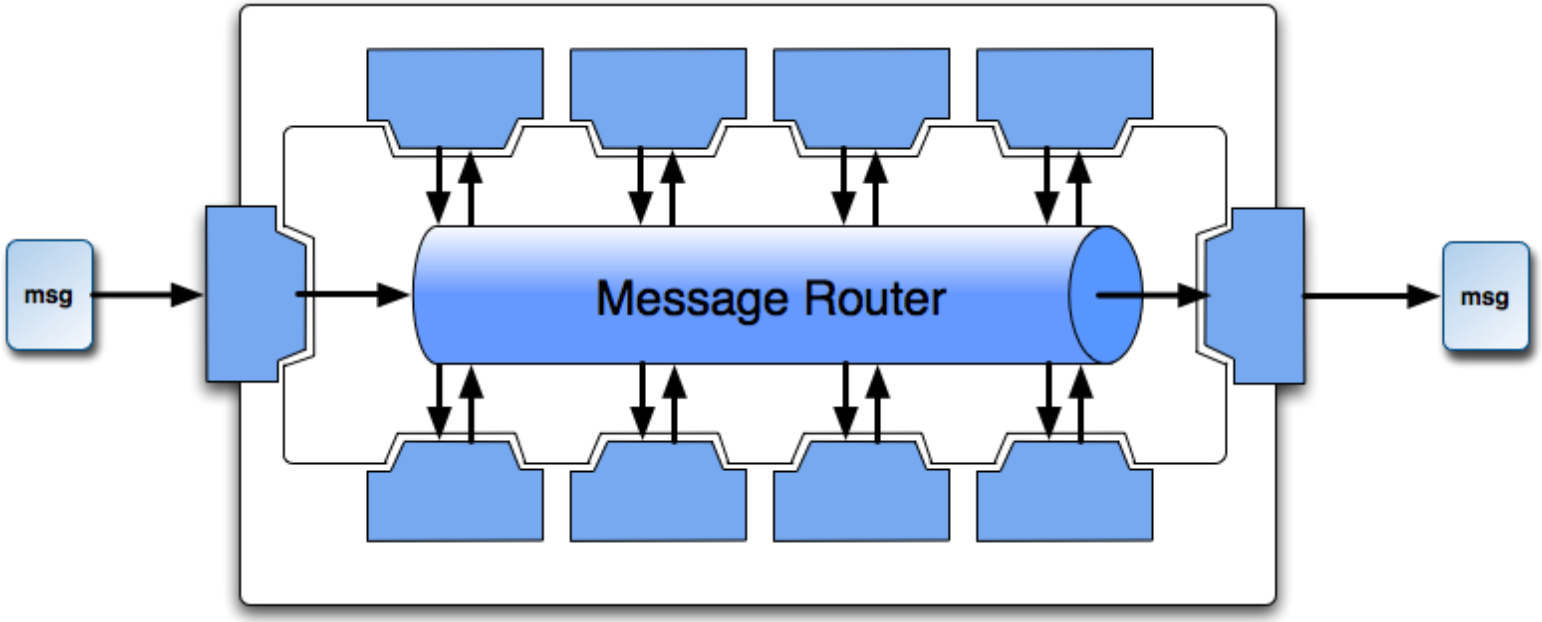
ServiceMix Binding Components

- Standard binding components
  - File – Polling and writing local files
  - FTP – Poll and send remote files
  - HTTP/S – Receive incoming requests, proxy to external services
  - JMS – Receive and send to queue / topic
  - VM – ActiveMQ VM transport for local queues
  - TCP – Use MINA for handling custom protocols
  - XMPP – Send and Receive from XMPP servers (Jabber, Wildfire, etc.)
- Custom development for proprietary systems

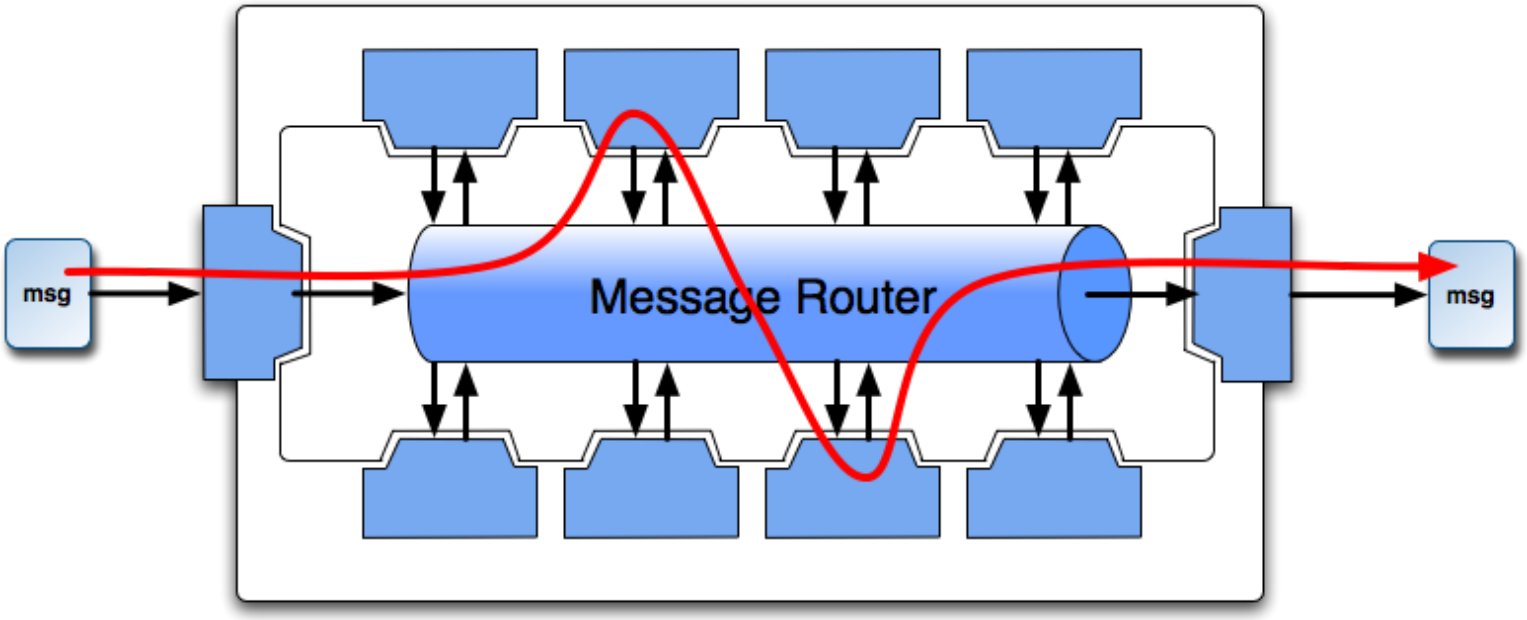
ServiceMix Service Engines

- Standard components
  - Camel – EIP routing and mediation
  - CXF – SOAP processing, WS-*
  - Drools – Business Rules
  - POJOs
  - Scripting – Groovy, JRuby, BSH, etc
  - XSLT – Transform XML
  - WS-Notification
- Custom service engines

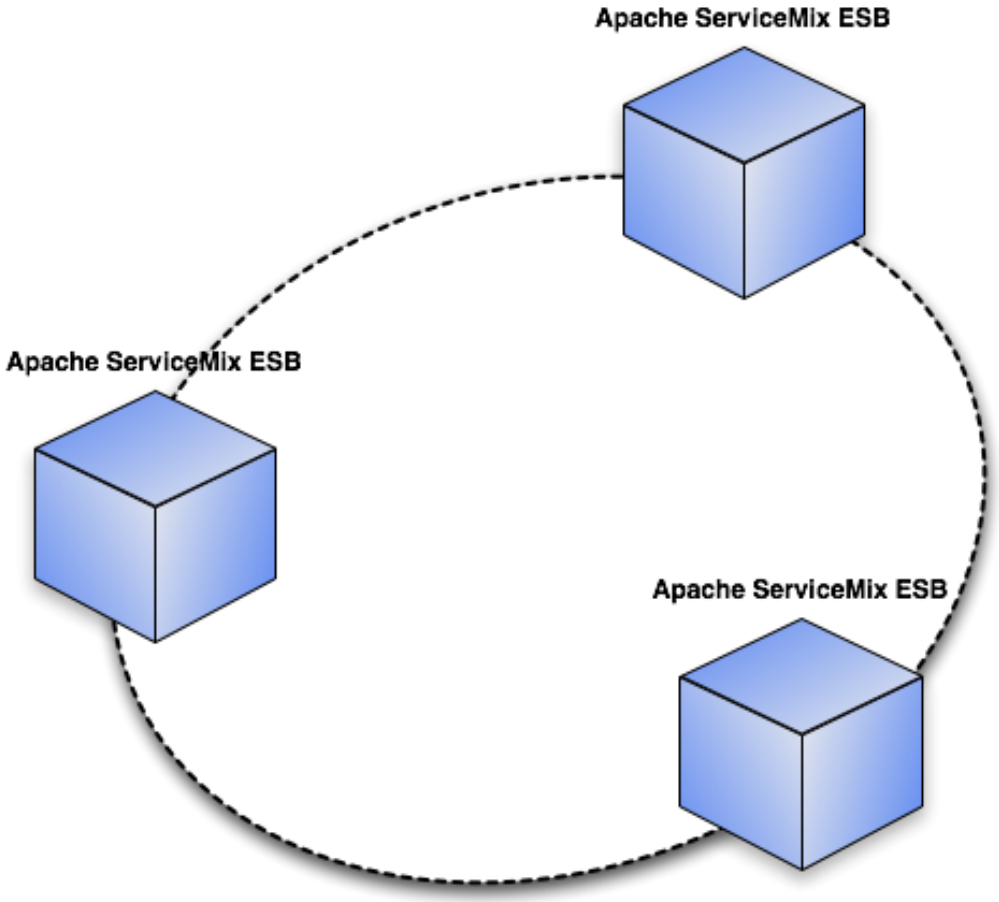
Message Routing



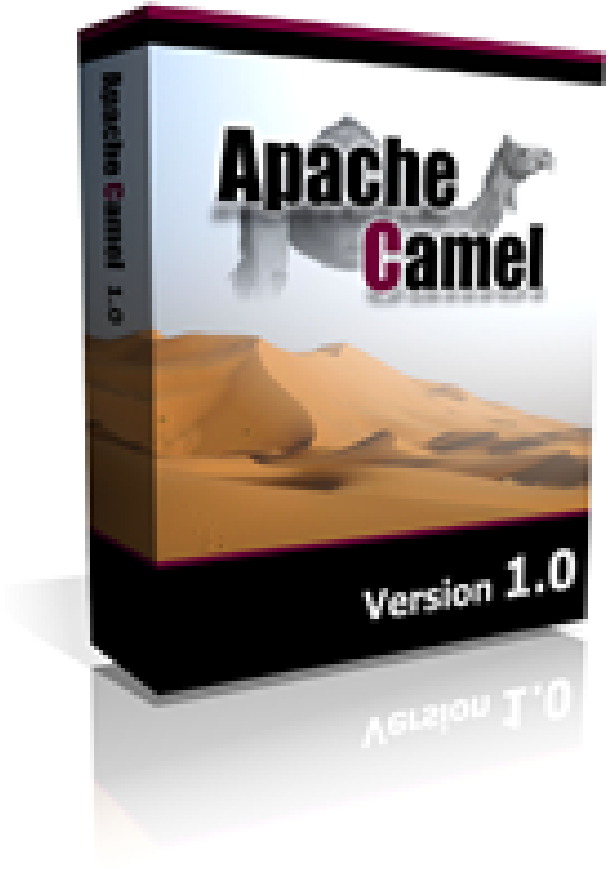
Message Routing



Distribution of ServiceMix Containers



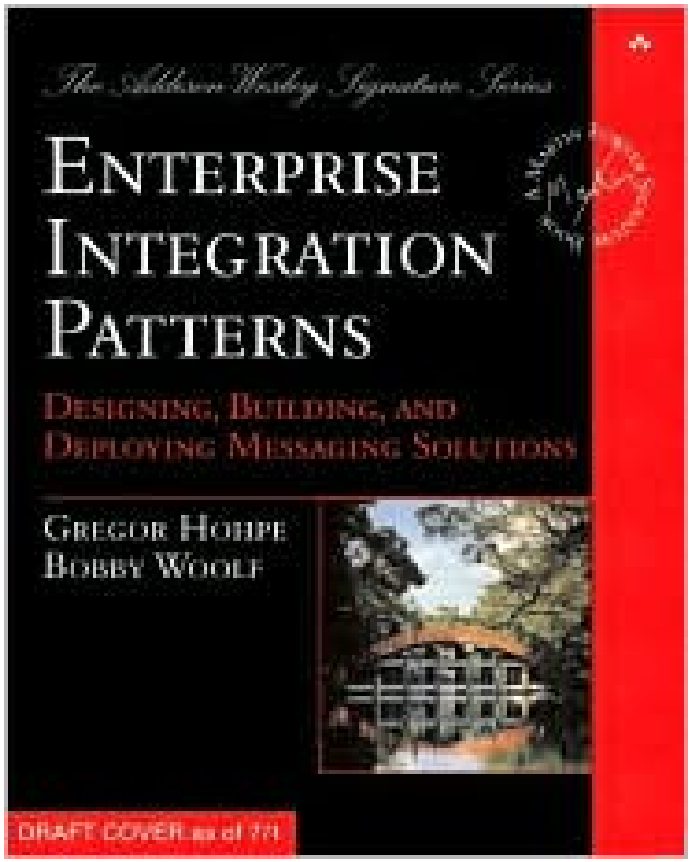
Introduction to Apache Camel



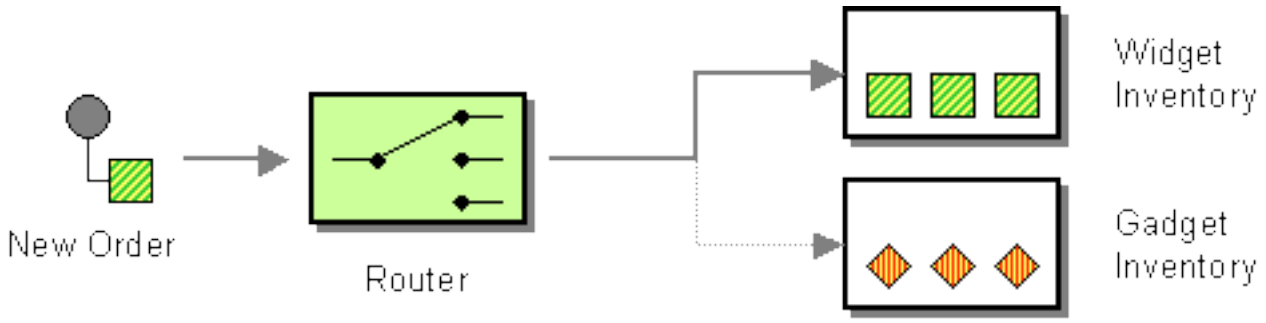
What is Camel?

- Implements Enterprise Integration Patterns
  - Common patterns in messaging and integration
- Concise language for defining routes
  - Java based DSL (syntax help in IDE)
  - Spring XML (XSD for easier editing)
- Simple URI strings to define endpoints
  - "file:src/test/data?noop=true"
  - Can be extended with custom URI
  - Extend with Spring POJO (uri string matches bean name)

What is EIP?



Example Pattern: Content Based Router

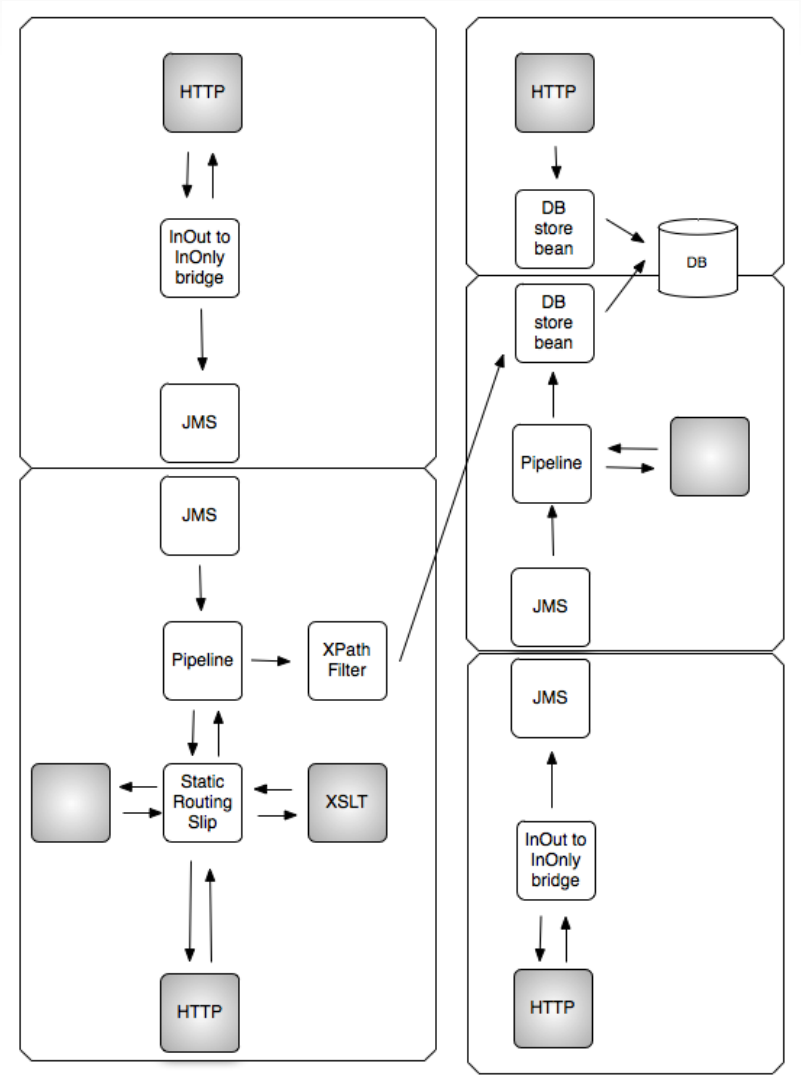



```
RouteBuilder builder = new RouteBuilder() {
    public void configure() {
        from("seda:a").choice().when(header("foo")
.isEqualTo("bar")).to("seda:b")
            .when(header("foo").isEqualTo("cheese"))
.to("seda:c").otherwise().to("seda:d");
    }
};
```

Example Pattern: Content Based Router

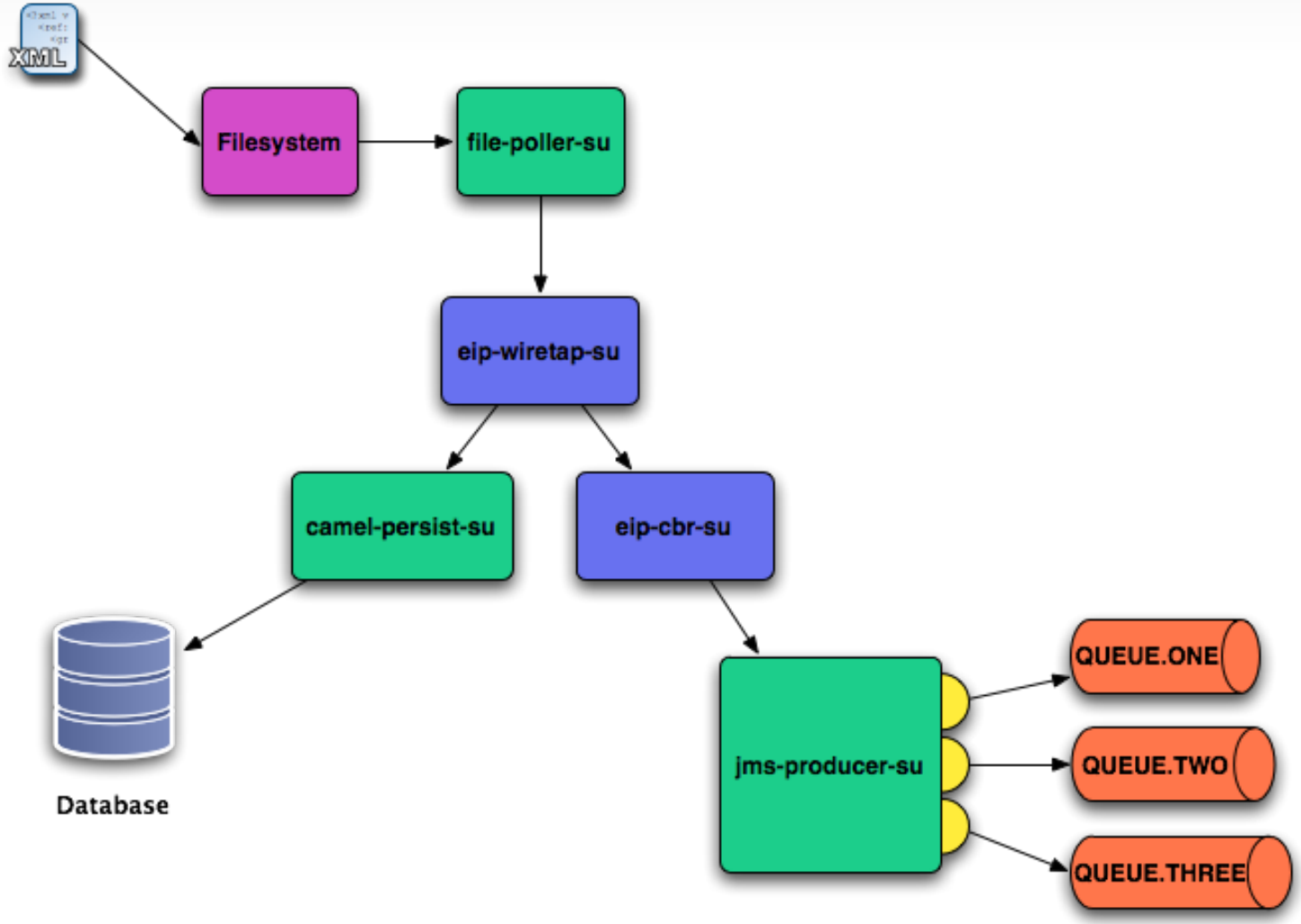
```
<camelContext id="buildSimpleRouteWithChoice"
    xmlns="http://activemq.apache.org/camel/schema/spring">
  <route>
    <from uri="seda:a"/>
    <choice>
      <when>
        <predicate>
          <header name="foo"/>
          <isEqualTo value="bar"/>
        </predicate>
        <to uri="seda:b"/>
      </when>
      <when>
        <predicate>
          <header name="foo"/>
          <isEqualTo value="cheese"/>
        </predicate>
        <to uri="seda:c"/>
      </when>
      <otherwise><to uri="seda:d"/></otherwise>
    </choice>
  </route>
</camelContext>
```

Camel Makes Routing Much Easier!




```
from("http://localhost:8080/requests/").
    tryBlock().
        to("activemq:queue:requests").
        setOutBody(constant("<ack/>")).
    handle(Throwable.class).
        setFaultBody(constant("<nack/>"));

from(("activemq:queue:requests?transacted=true").
    process(requestTransformer).
    to("http://host:8080/Request").
    filter(xpath("//nack")).
    process(nackTransformer).
    to("jdbc:store");

from("http://localhost:8080/responses/").
    tryBlock().
        to("activemq:queue:responses").
        setOutBody(constant("<ack/>")).
    handle(Throwable.class).
        setFaultBody(constant("<nack/>"));

from("activemq:queue:responses?transacted=true").
    process(responseTransformer).
    to("jdbc:store");

from("http://localhost:8080/pull/").
    to("jdbc:load");
```

ServiceMix and Camel Working Together



Configuration

file-poller-su

```
<beans xmlns:file='http://servicemix.apache.org/file/1.0'
       xmlns:myApp="http://com.mycompany/myapp">

  <file:poller service="myapp:file"
               endpoint="poller"
               file="file:/var/filedrop/inbox"
               targetService="myapp:wiretap"
               targetEndpoint="logger" />
</beans>
```

eip-wiretap-su

```
<beans xmlns:eip="http://servicemix.apache.org/eip/1.0"
       xmlns:myapp="http://mycompany.com/myapp">

  <eip:wire-tap service="myapp:wiretap" endpoint="endpoint">
    <eip:target>
      <eip:exchange-target service="myapp:cbr" />
    </eip:target>
    <eip:inListener>
      <eip:exchange-target service="myapp:persist" />
    </eip:inListener>
  </eip:wire-tap>

</beans>
```

camel-persist-su

```
public class PersistOrderRoute extends RouteBuilder {
    public void configure() {
   from("jbi:endpoint:http://mycompany.com/persist/order")
      .convertBodyTo(Order.class)
      .to("jpa:com.mycompany.Order?persistenceUnit=order-proc")
      .convertBodyTo(Order.class);
    }}
```

```
<beans xmlns:camel=”http://activemq.apache.org/camel">
  <camel:camelContext id="camel">
    <package>com.mycompany.persistence</package>
  </camel:camelContext>
</beans>
```

eip-cbr-su

```
<beans xmlns:eip="http://servicemix.apache.org/eip/1.0"
       xmlns:myapp="http://mycompany.com/myapp">

  <eip:content-based-router service="myapp:cbr"
    endpoint="endpoint">
    <eip:rules>
      <eip:routing-rule>
        <eip:predicate>
          <eip:xpath-predicate
            xpath="/message/cheese/text() = 'gouda'" />
        </eip:predicate>
        <eip:target>
          <eip:exchange-target service="myapp:queue1" />
        </eip:target>
      </eip:routing-rule>
...
```

eip-cbr-su

```
...
    <eip:routing-rule>
      <eip:predicate>
        <eip:xpath-predicate
          xpath="/message/cheese/text() = 'swiss'" />
      </eip:predicate>
      <eip:target>
        <eip:exchange-target service="myapp:queue2" />
      </eip:target>
    </eip:routing-rule>
    <eip:routing-rule>
      <eip:target>
        <eip:exchange-target service="myapp:queue3" />
      </eip:target>
    </eip:routing-rule>
  </eip:rules>
</eip:content-based-router>
```

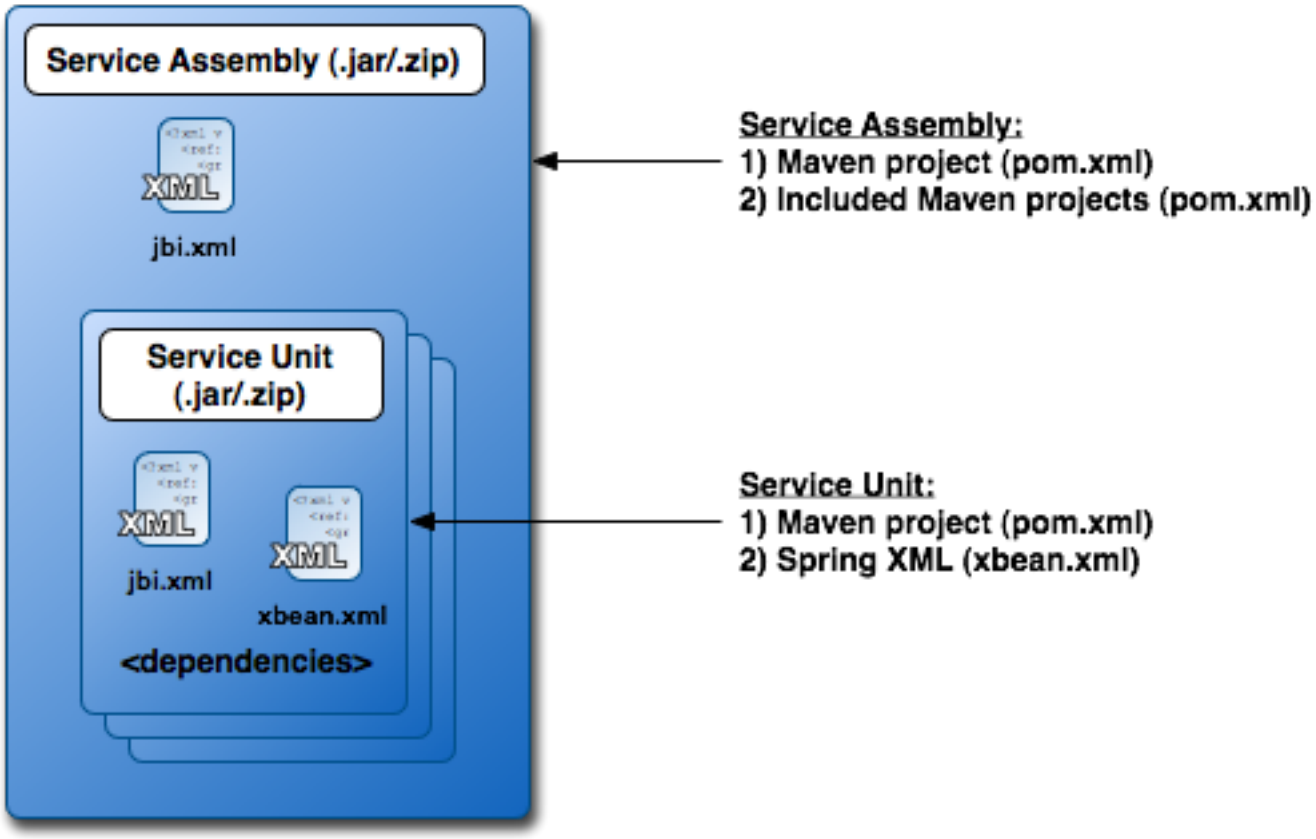
jms-producer-su

```
<beans xmlns:jms="http://servicemix.apache.org/jms/1.0"
    xmlns:myapp="http://mycompany.com/myapp
    xmlns:amq="http://activemq.org/config/1.0">

  <jms:endpoint service="myapp:queue1"
            endpoint="myProvider"
            role="provider"
            destinationStyle="queue"
            jmsProviderDestinationName="queue1"
            connectionFactory="#connectionFactory" />

  <jms:endpoint service="myapp:queue2"
            endpoint="myProvider"
            role="provider"
            destinationStyle="queue"
            jmsProviderDestinationName="queue2"
            connectionFactory="#connectionFactory"/>
```

jms-producer-su

```
...
   <jms:endpoint service="myapp:queue3"
                 endpoint="myProvider"
                 role="provider"
                 destinationStyle="queue"
                 jmsProviderDestinationName="queue3"
                 connectionFactory="#connectionFactory"/>

   <amq:connectionFactory id="connectionFactory"
    brokerURL="tcp://localhost:61616" />

</beans>
```

JBI Packaging



ServiceMix or Camel?

- ServiceMix
  - Full Container for integration applications
  - Distributed environment
  - Standardized
- Camel
  - Embeddable – Small footprint, Spring friendly
  - Standalone

Apache ServiceMix 4.0

Building Blocks

- Runtime: OSGi (Apache Felix)
- Message Broker: Apache ActiveMQ
- SOAP Support: Apache CXF
- Routing Engine: Apache Camel

Fuse Overview

- Supported builds
- Development support
- Production support
- Value added – Tooling and Management

Links

- http://servicemix.apache.org
- http://activemq.apache.org/camel
- http://jcp.org/aboutJava/communityprocess/final/jsr208
- http://open.iona.com

Making Software Work Together™