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

ServiceMix

Building a Service Oriented Architecture with ServiceMix

Jeff Genender
CTO
Savoir Technologies, Inc

SAVOIR TECHNOLOGIES

Colorado Avalanche


THE WORLD

Alaska

My place in Colorado

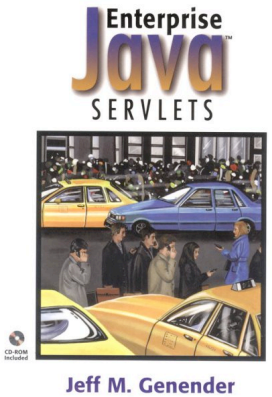
My expectation of Sweden

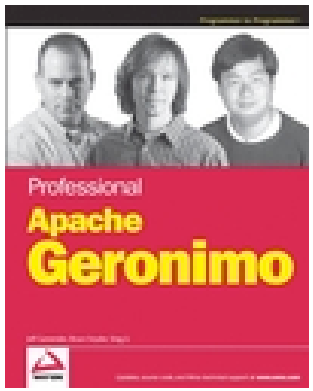
This is what I got

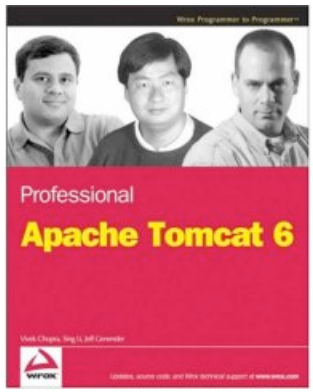
Jeff Genender

Apache CXF

JSR 316 - Java EE 6



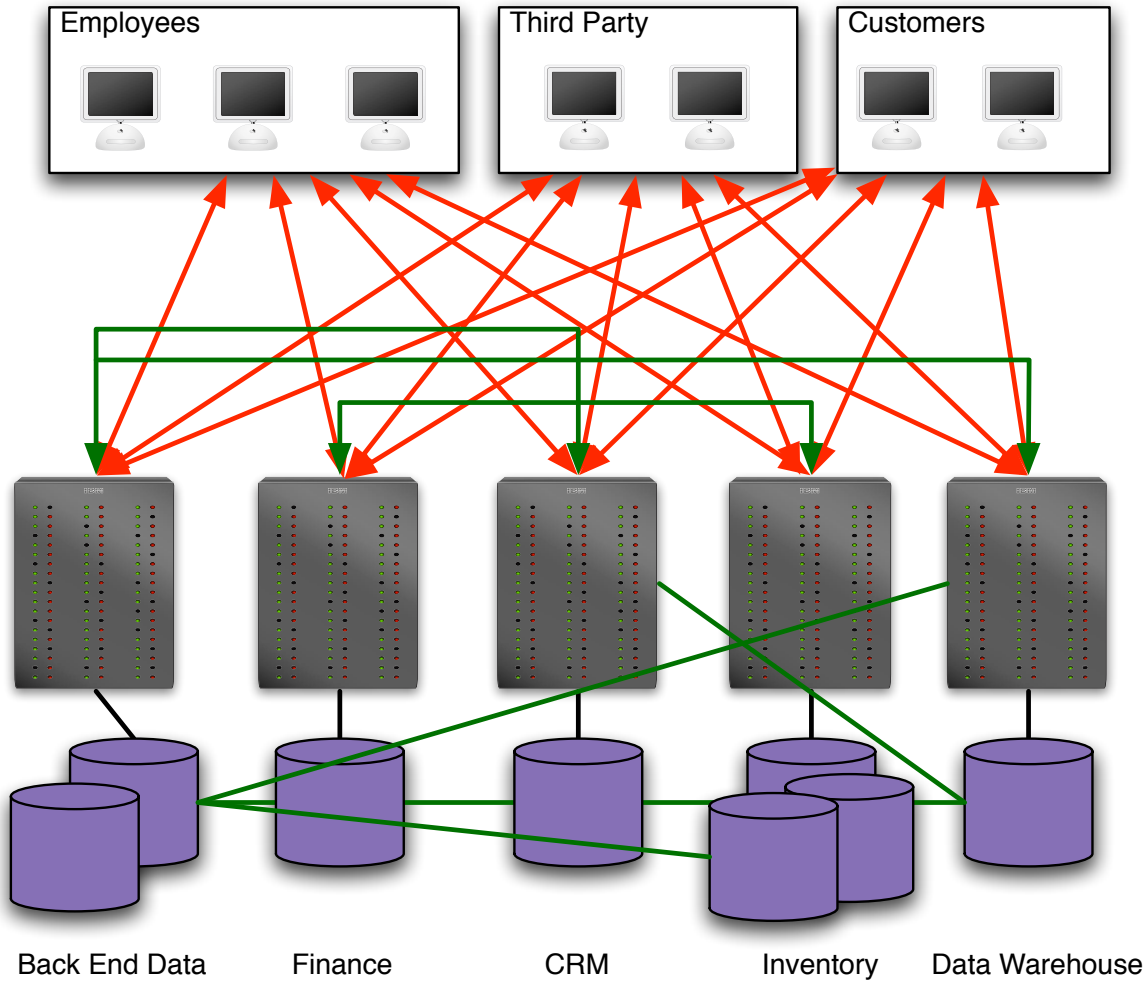




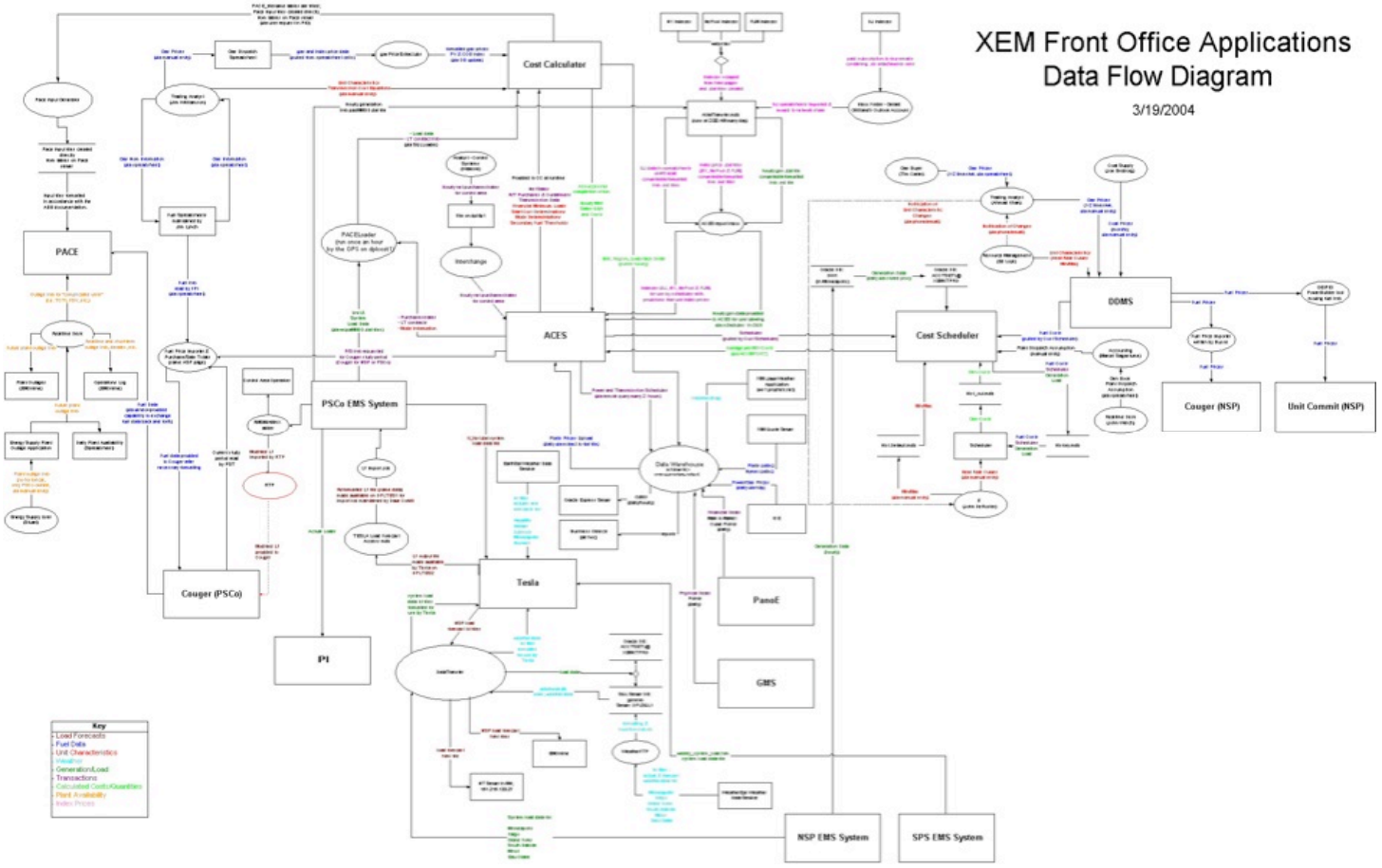
Agenda

- Business Integration Problem
  - The Enterprise Service Bus (ESB)
- JBI - Java Business Integration
- ServiceMix
  - Overview
  - Hands-On

Business Integration Problem

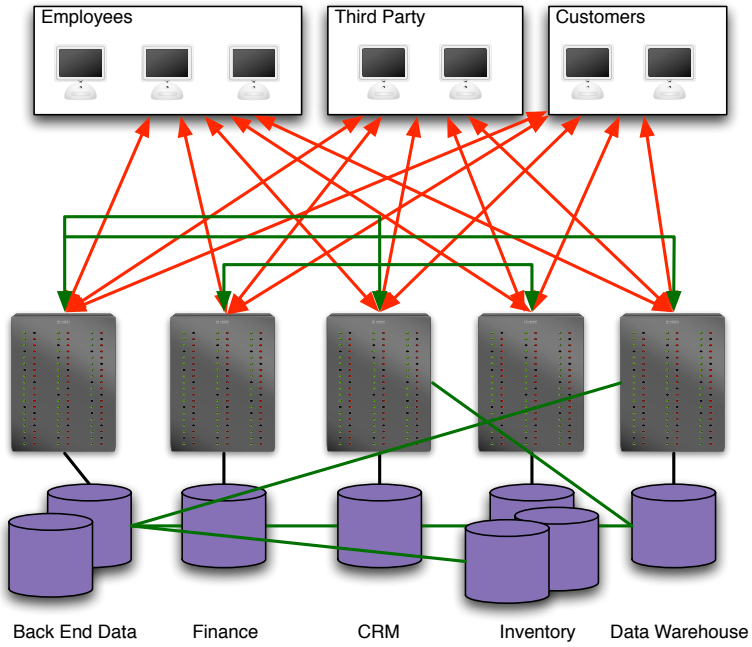


Business Integration Problem



Service Oriented Architecture

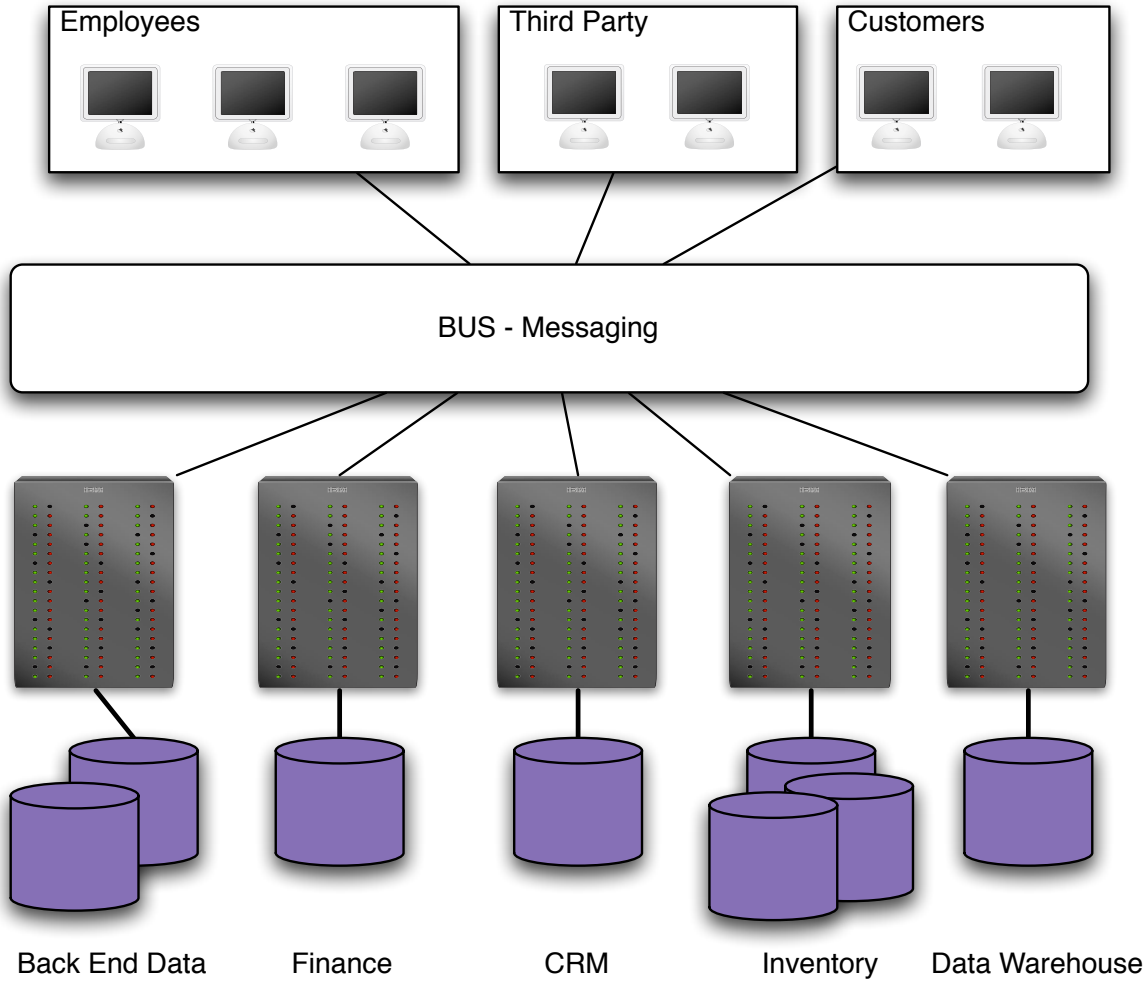
- Interoperability
- Reusable Components
- Language and Platform Neutral
- Open Standards
- Standards Based
- Encapsulation
- Loose Coupling



Enterprise Service Bus (ESB)

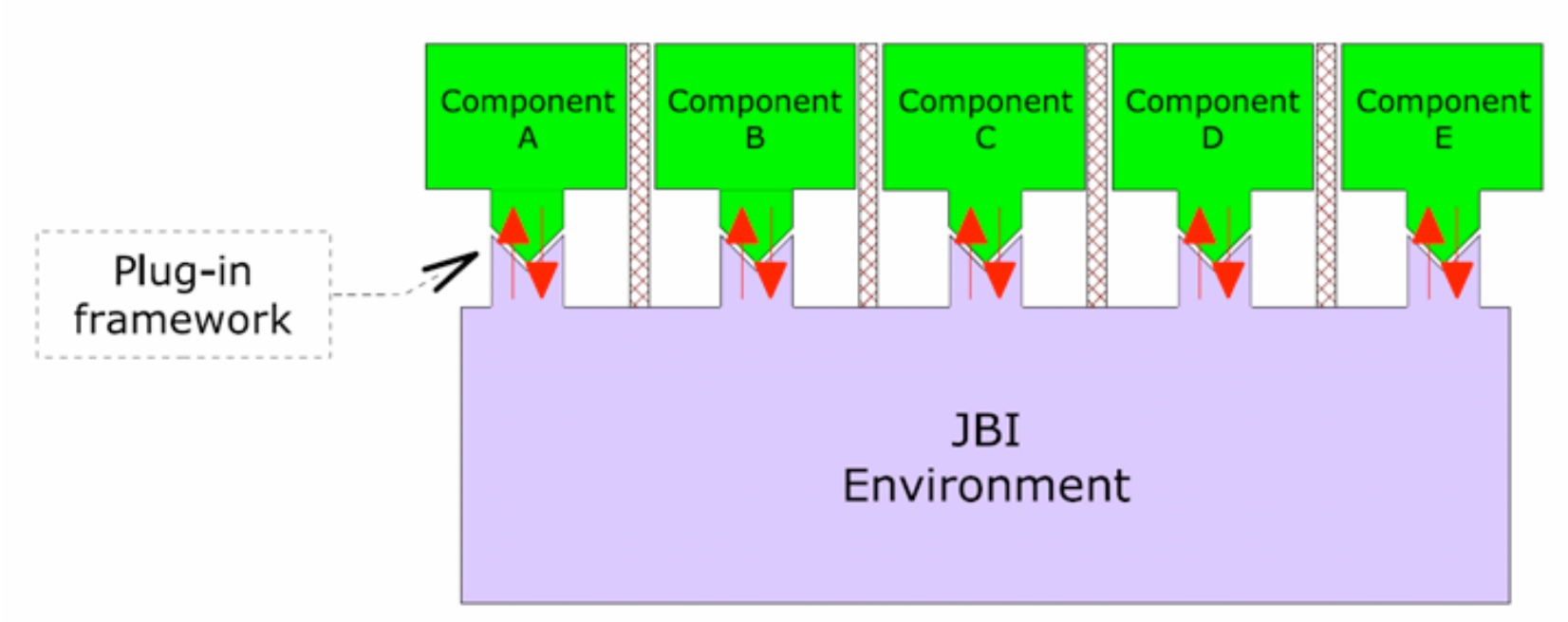
- All facets of a SOA but more...
  - Message Based Routing
    - Message Oriented Middleware - Components know their messages
    - Reliable (JMS)
  - Orchestration
    - Workflow
    - Transactionality
    - Routing
  - Transformation
    - Converting data from one type to another

Enterprise Service Bus (ESB)

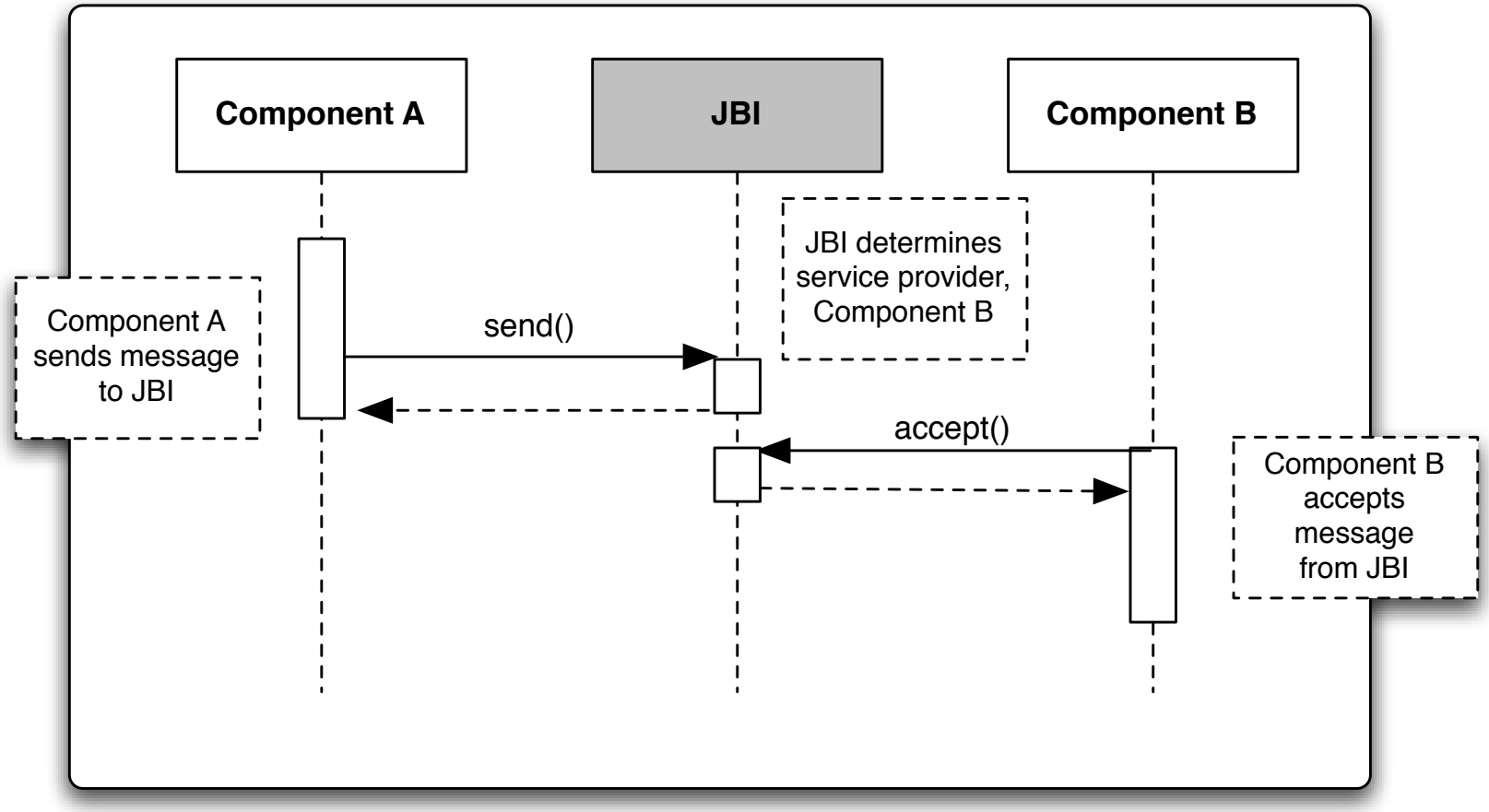


Java Business Integration (JBI - JSR 208)

- Open Standard to Define a Pluggable Architecture
  - Components are responsible for consuming and providing services
  - Components plug into the JBI framework
  - Third party products “just work”



JBI Decoupling



Java Business Integration (JBI - JSR 208)

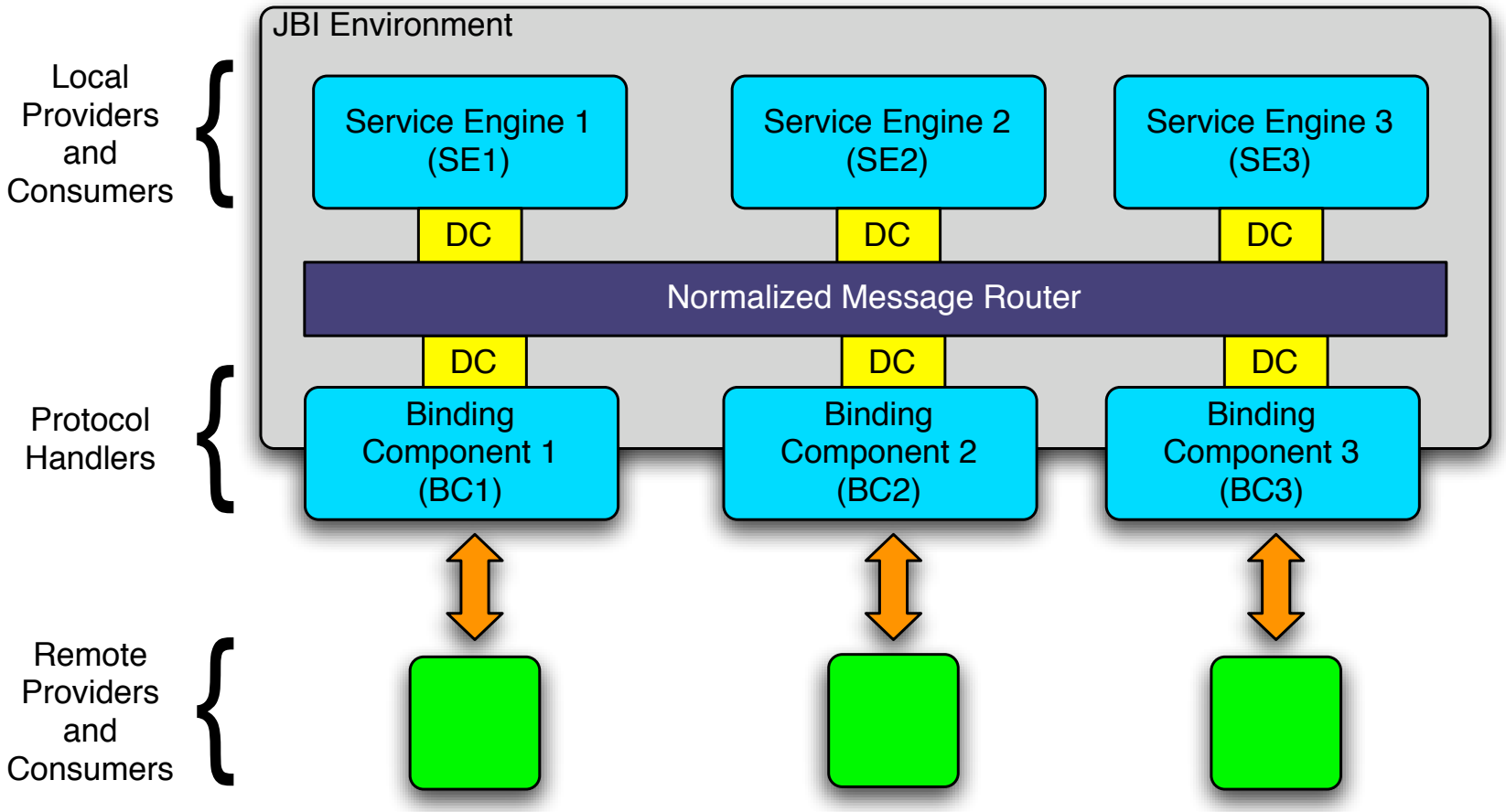
- 2 types of components
  - Service Engine (SE) - Provides business logic and transformation
  - Binding Component (BC) - Provides connectivity to external JBI
- Messaging
  - Web Services Description Language (WSDL) Based Services (2.0/1.1)
  - Normalized Messages
    - All messages must be “Normalized” before they enter the bus
    - All messages are de-normalized when exiting via Binding Component
    - All components can get a common message format
    - 3 parts, Payload, Message Properties, Message attachments

Java Business Integration (JBI - JSR 208)

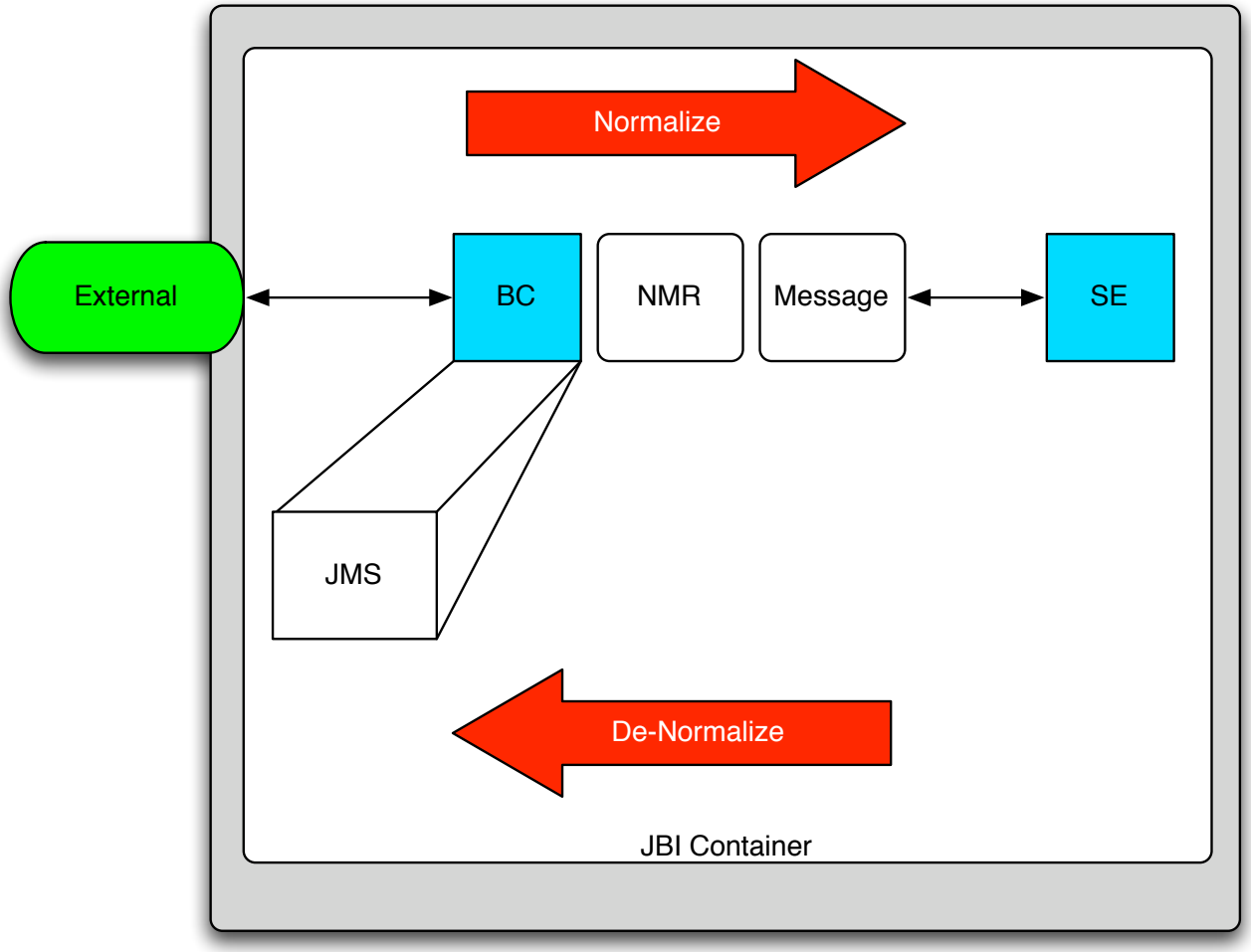
- Normalized Message Router (NMR Architecture)
  - Receives and exchanges messages from JBI components
  - Routes the messages to the proper component for processing
- Message Exchange Patterns (MEPS) (from WSDL 2.0)
  - In-Only
  - Robust-only
  - In-Out
  - In Optional-Out

- Delivery Channel (DC) - The actual communication pipe
  - Kind of like a socket

JBI up close


JBI Environment
Local Providers and Consumers
Service Engine 1 (SE1)
Service Engine 2 (SE2)
Service Engine 3 (SE3)
DC
DC
DC
Normalized Message Router
DC
DC
DC
Protocol Handlers
Binding Component 1 (BC1)
Binding Component 2 (BC2)
Binding Component 3 (BC3)
Remote Providers and Consumers

JBI Up Close Messaging

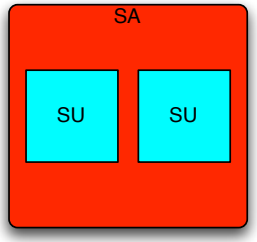


JBI Deployment Artifacts

- Container Based
  - Binding Components
  - Service Engines
- Application Based
  - Service Unit (SU)
  - Service Archive (SA)

JBI Packaging - zip or jar files - 4 types

- Components
  - Installer containing a component that transforms or communicates (SE/BC)
- Shared Libraries
  - Collection of jars that can be shared by several components
- Service Units (SU)
  - Configuration for a JBI component
- Service Assemblies (SA)
  - Collection of Service Units
  - Like an EAR for JBI Service Units

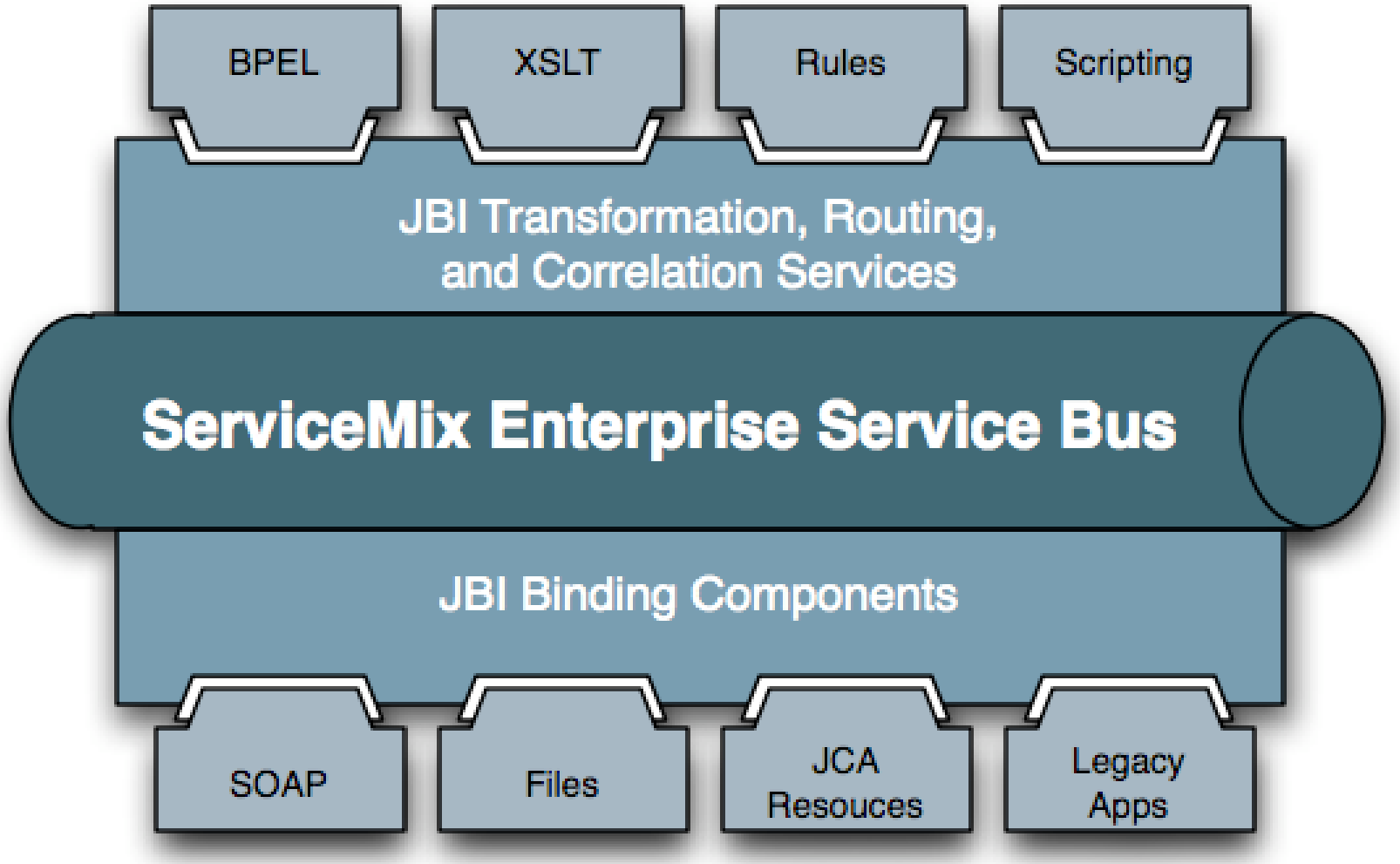


Inside a JBI Component or SA

- Its a zip or jar
- META-INF/jbi.xml - The deployment descriptor
- lib - dependency jars and SU, BC, and SE

```
<?xml version="1.0" encoding="UTF-8"?>
<jbi xmlns="http://java.sun.com/xml/ns/jbi" version="1.0">
  <component type="service-engine"
             component-class-loader-delegation="parent-first"
             bootstrap-class-loader-delegation="parent-first">
    <identification>
      <name>servicemix-eip</name>
      <description>ServiceMix :: EIP</description>
    </identification>
    <component-class-name>org.apache.servicemix.eip.EIPComponent</component-class-name>
    <component-class-path>
      <path-element>lib/servicemix-eip-3.1-incubating.jar</path-element>
    </component-class-path>
    <bootstrap-class-name>org.apache.servicemix.eip.EIPBootstrap</bootstrap-class-name>

    <bootstrap-class-path>
        <path-element>lib/servicemix-eip-3.1-incubating.jar</path-element>
    </bootstrap-class-path>
    <shared-library version="3.1-incubating">servicemix-shared</shared-library>
  </component>
```

Apache ServiceMix - The Open Source ESB



ServiceMix Marketing Shpiel

- Messaging Bus - ActiveMQ
- Embeddable
  - Container can be embedded in other applications
    - Apache Geronimo
  - Great during development
    - Run your own small version in IDEs
- Many instances of ServiceMix can be tied together
  - All instances will know where all of the components live
  - Done through JMS
  - Great for major corporate ESB initiatives

ServiceMix Included Service Engines

- servicemix-bean - Exposes pojos
- servicemix-drools - JBI integration into the Drools engine
- servicemix-camel - Routing container based on Apache camel
- servicemix-cxf-se - Certified JAXWS component
- servicemix-eip - Routing container
  - Based on Gregor Hohpe's Enterprise Integration Patterns book
- servicemix-file - Read/write files or poll for files
- servicemix-jsr181 - Exposes annotated web services on the bus
- servicemix-lwcontainer - Lightweight components
- servicemix-quartz - Schedule and trigger jobs using Quartz Scheduler
- servicemix-saxon - XSLT processing and XQuery

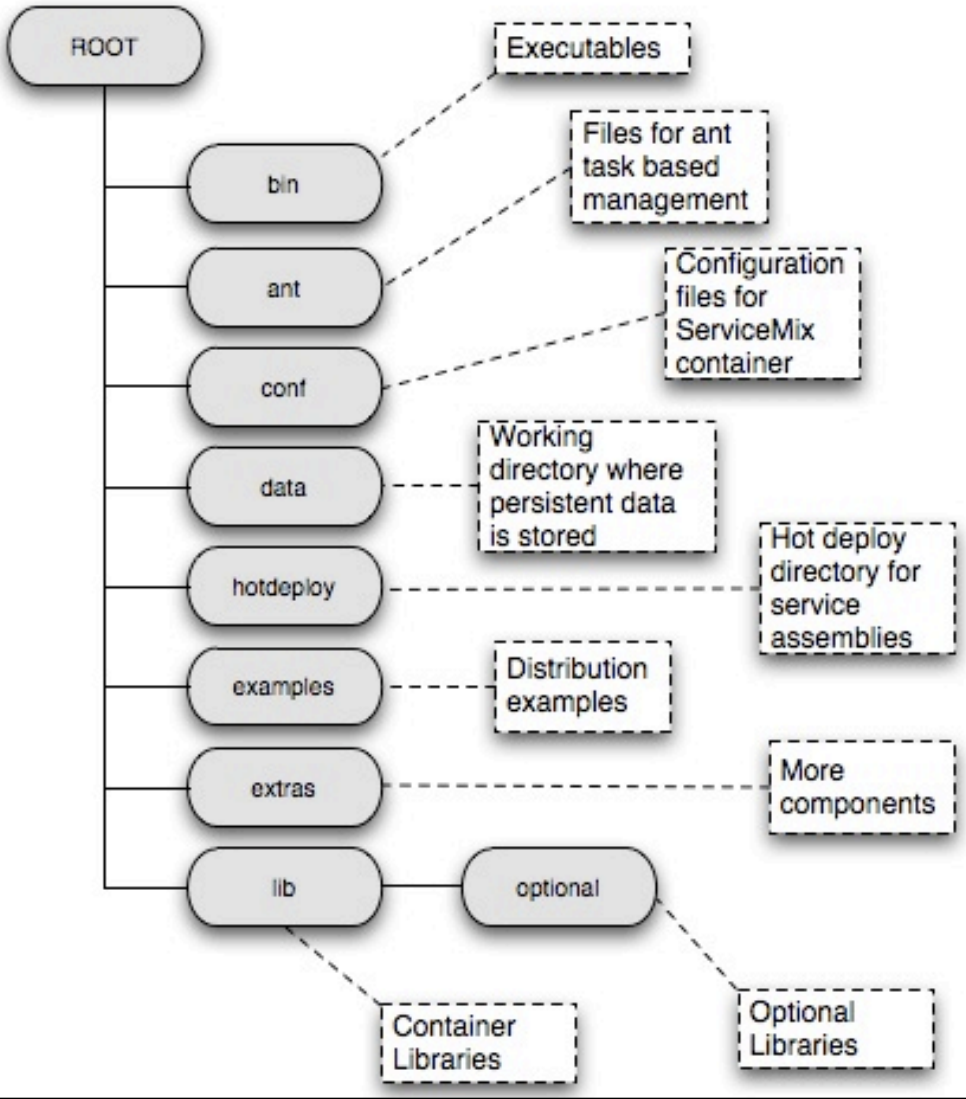
ServiceMix Included Service Engines

- servicemix-script - Provides JBI integration with scripting engines.
  - groovy
  - Any JSR-223 compliant scripting language
- servicemix-wsn2005
  - Implements the WS-Notification specification from Oasis.

ServiceMix Included Binding Components

- servicemix-ftp - Provides integration to FTP
- servicemix-http - Provide http integration and access
- servicemix-email - Email to/from components
- servicemix-cxf-bc - SOAP/HTTP conduit
- servicemix-jms - Allow for Topic/Queue messaging over the bus
- servicemix-xmpp - Communication via the Jabber protocol
- servicemix-rss - Can interface with RSS feeds and clients

ServiceMix Directory Layout



Developing with ServiceMix

- Uses XBean to wire together components
  - Spring on steroids
- 80-90% is wiring current BCs and SEs
  - Most BCs and SEs will do what you want
  - SUs might contain some code or they might not
- Each component may have its own XBean syntax

Spring Syntax

```
<beans>
  <bean id="jbi" class="org.apache.servicemix.jbi.container.SpringJBIContainer">
    <property name="embedded" value="true" />
    <property name="broker">
      <bean class="org.apache.servicemix.jbi.security.SecuredBroker">
        <property name="authorizationMap" ref="authorizationMap" />
        <property name="flows">
          <list>
            <bean class="org.apache.servicemix.jbi.flows.seda.SedaFlow" />
            <bean class="org.apache.servicemix.jbi.flows.jms.JMSFlow">
              <property name="jmsURL" value="tcp://localhost:61616" />
            </bean>
            <bean class="org.apache.servicemix.jbi.flows.jca.JCAFlow">
              <property name="jmsURL" value="tcp://localhost:61616" />
              <property name="connectionManager" ref="connectionManager" />
            </bean>
          </list>
        </property>
      </bean>
    </property>
  </bean>
</beans>
```

XBean Syntax

```
<beans xmlns:sm="http://servicemix.apache.org/config/1.0">
  <sm:container id="jbi" embedded="true">
    <sm:broker>
      <sm:securedBroker authorizationMap="#authorizationMap">
        <sm:flows>
          <sm:sedaFlow />
          <sm:jmsFlow jmsURL="tcp://localhost:61616" />
          <sm:jcaFlow connectionManager="#connectionManager"
                      jmsURL="tcp://localhost:61616" />
        </sm:flows>
      </sm:securedBroker>
    </sm:broker>
  </sm:container>
</beans>
```

Example SU wiring a File Service Engine

```
<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:/Users/jgenender/poller/inbox"
              targetService="myapp:wiretap"
              targetEndpoint="logger" />
</beans>
```

Example SU wiring a HTTP Binding Component

```
<beans xmlns:http="http://servicemix.apache.org/http/1.0"
       xmlns:person="http://servicemix.apache.org/samples/wsdl-first">

  <http:endpoint service="person:PersonService"
                 endpoint="soap"
                 targetService="person:PersonService"
                 role="consumer"
                 locationURI="http://0.0.0.0:8192/PersonService/"
                 defaultMep="http://www.w3.org/2004/08/wsdl/in-out"
                 soap="true" />

</beans>
```

Developing in ServiceMix

- Ask yourself...do you REALLY need to write a SC or BC?
- Likely will write SUs and wrap with SA
- You can write everything by hand and use ant or...
- Use the maven archetypes with the jbi-maven-plugin

```
mvn archetype:create \
    -DarchetypeGroupId=org.apache.servicemix.tooling \
    -DarchetypeArtifactId=servicemix-archetype-name \
    -DarchetypeVersion=SM-ARCHETYPE-VERSION \
    -DgroupId=org.apache.servicemix.samples.embedded \
    -DartifactId=servicemix-embedded-example
    -DremoteRepositories=http://people.apache.org/repo/m2-incubating-repository
```

- Archtypes for just about all BC and SE (and SA, etc)

Maven Archetypes

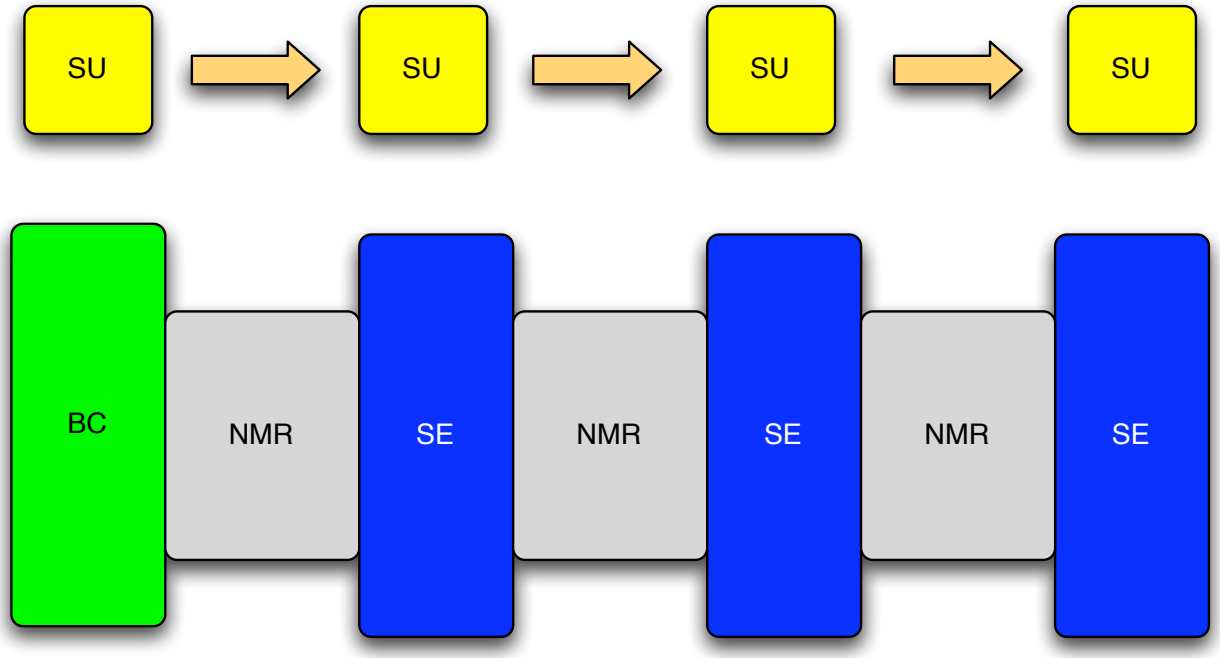
- servicemix-binding-component
- servicemix-service-engine
- servicemix-service-unit
- servicemix-service-assembly
- servicemix-shared-library
- servicemix-ftp-poller-service-unit
- servicemix-ftp-sender-service-unit
- servicemix-http-consumer-service-unit
- servicemix-http-provider-service-unit
- servicemix-jms-consumer-service-unit
- servicemix-jms-provider-service-unit
- servicemix-jsr181-wsdl-first-service-unit
- servicemix-lwcontainer-service-unit
- servicemix-camel-service-unit
- servicemix-eip-service-unit
- servicemix-embedded-simple
- servicemix-ode-service-unit
- servicemix-jsr181-annotated-service-unit
- servicemix-saxon-xquery-service-unit
- servicemix-saxon-xslt-service-unit
- servicemix-bean-service-unit
- servicemix-drools-service-unit
- servicemix-archetypes-itests
- servicemix-cxf-bc-service-unit
- servicemix-cxf-se-service-unit

Developing in ServiceMix

- Write an SU and wrap it in a SA
- Write some supporting code if needed
- XBean used for wiring and will reside in the root of your jar
- XBean has specific syntax what what you are wiring
  - service, endpoint, role, locationURI, defaultMep

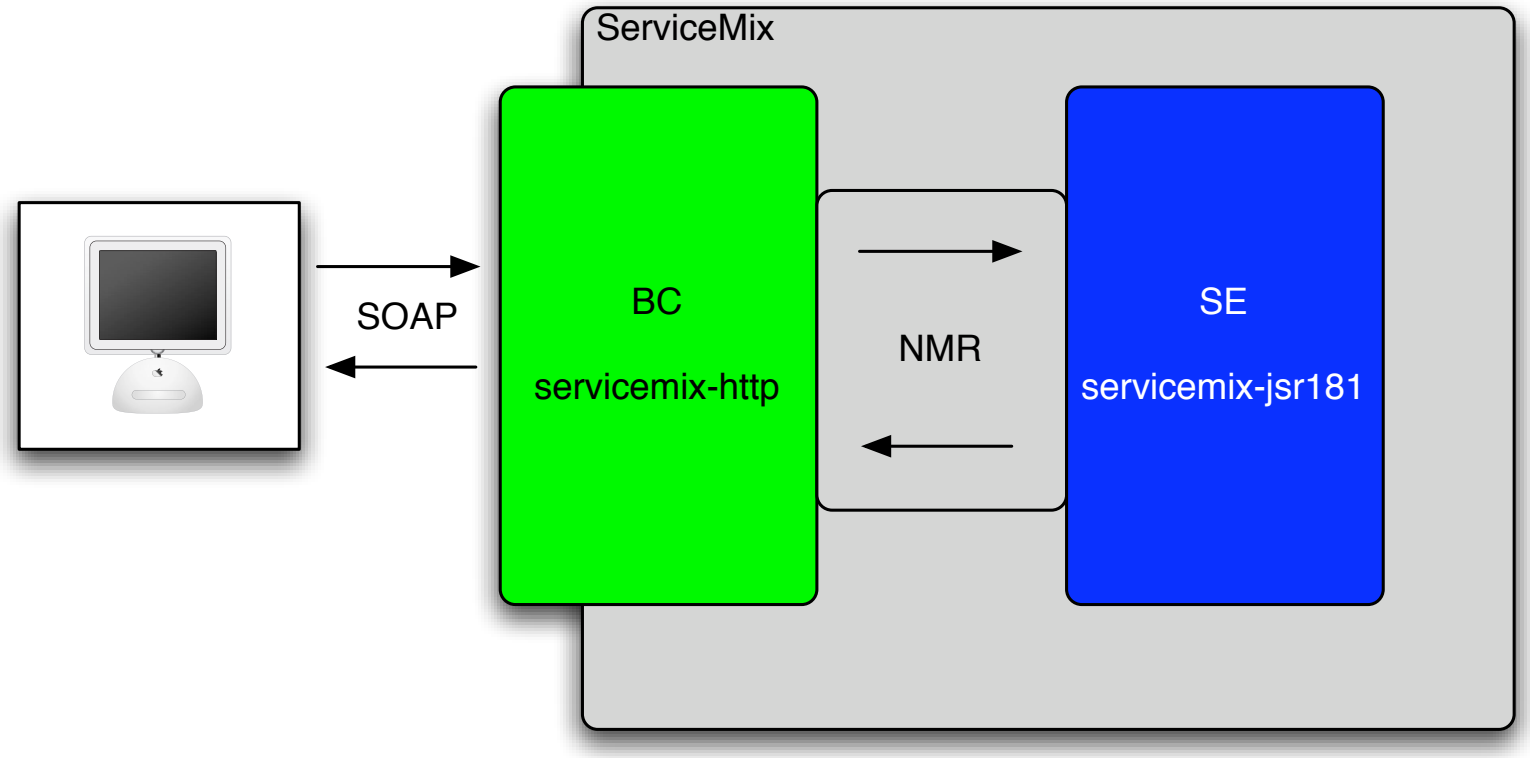
```
<beans xmlns:http="http://servicemix.apache.org/http/1.0"
       xmlns:person="http://servicemix.apache.org/samples/wsdl-first">

  <http:endpoint service="person:PersonService"
                 endpoint="soap"
                 role="consumer"
                 locationURI="http://0.0.0.0:8192/PersonService/"
                 defaultMep="http://www.w3.org/2004/08/wsdl/in-out"
                 soap="true"
                 soapAction="getPerson"/>

</beans>
```

Developing in ServiceMix

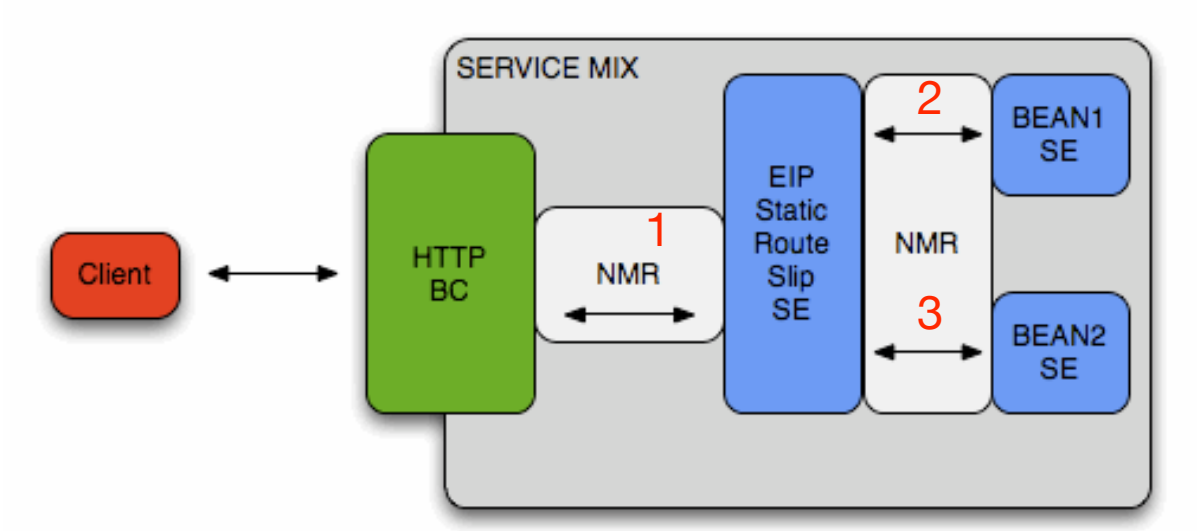


DEMO

WSDL First Example



Transformation Example - More Complex



Questions