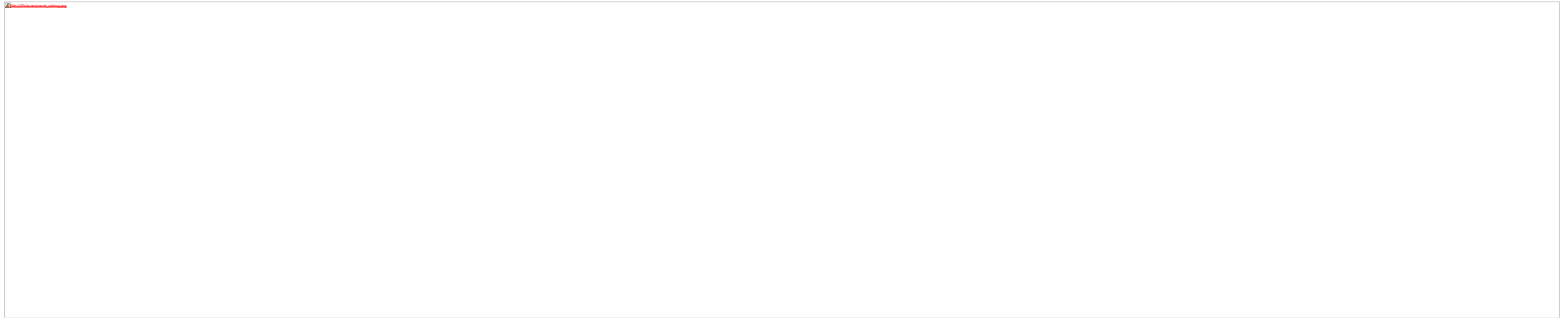


# Next Generation Open Source SOA

Burr Sutter  
Sr. Product Manager, SOA  
(JBossESB, Riftsaw, jBPM, Drools)  
September 3, 2009



# Agenda

- Is SOA Dead?
- JBoss SOA Platform Overview & Roadmap
- RESTful inclinations
- Complex Event Processing (CEP) w/ Drools Fusion
- Infinispan + ESB
- BPEL

## Is SOA Dead?

"The report of my death was  
an exaggeration."

Mark Twain

# SOA, ROA & WOA – Oh My!

- **ROA – Resource Oriented Architecture - REST**  
(Representational State Transfer – Roy Fielding)
- **WOA** – “Web Oriented Architecture (WOA) is a style of software architecture that extends service-oriented architecture (SOA) to web based applications, and is sometimes considered to be a light-weight version of SOA. WOA is also aimed at maximizing the browser and server interactions by use of technologies such as REST and POX.” - Wikipedia
- **1<sup>st</sup> SOA is NOT SOAP**  
- architectural style vs wire protocol
- **2<sup>nd</sup> SOA is NOT WS-\***  
- patterns & principles vs a large body of standards

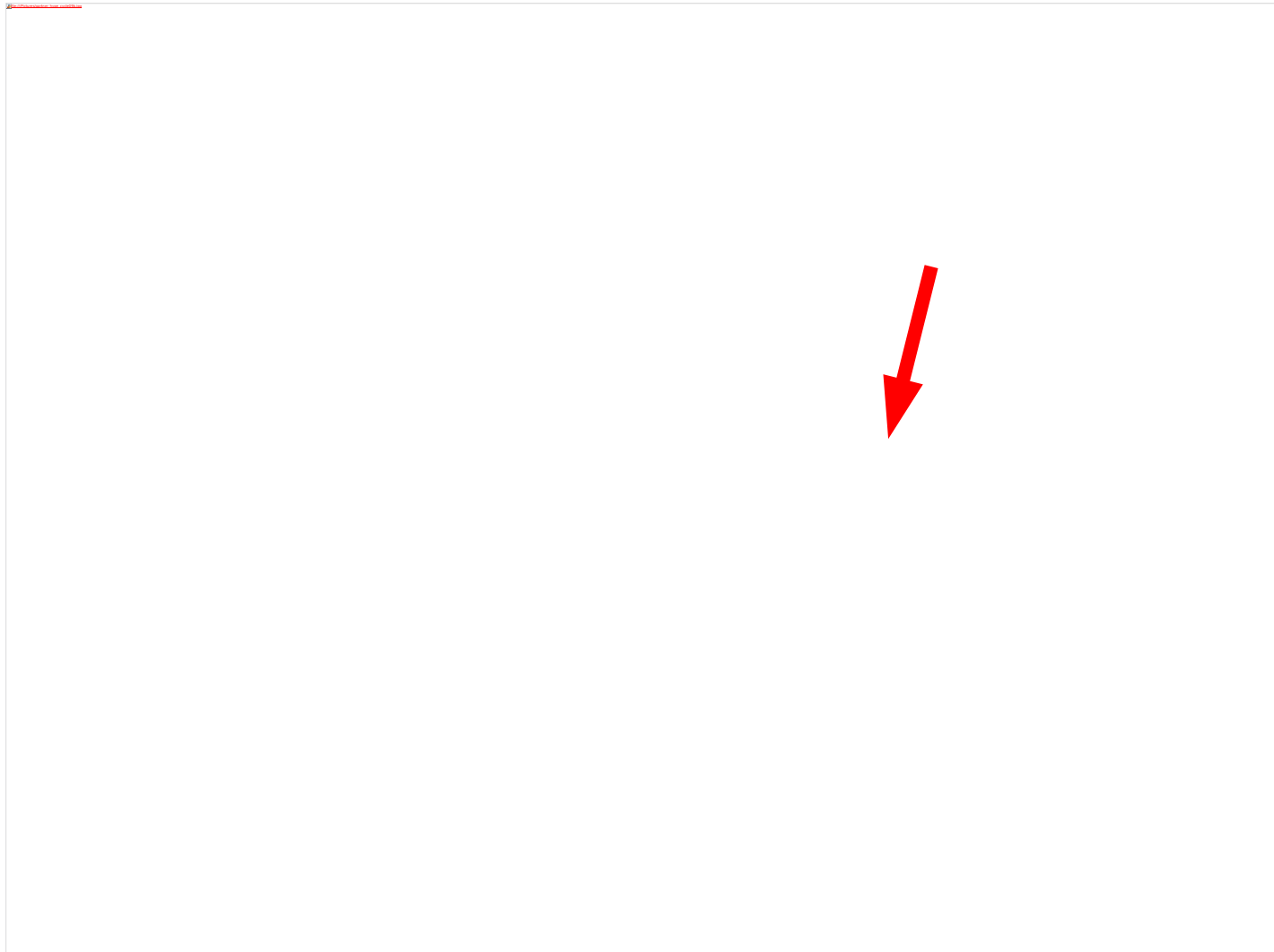
# SOAP vs REST Technical Differences

- HTTP Verbs:  
SOAP – POST  
REST – GET, PUT, POST, DELETE
- Contract Definition:  
SOAP – WSDL – operations & messages  
REST – HTTP Verbs are the operations, messages may or may not have a contract via XSD and/or JSON (POX – plain 'ol XML)
- Content-Type:  
SOAP – focused on XML  
REST – allows for any payload – XML, JSON, Atom, RSS, etc.
- Clients:  
SOAP – requires a client that understands WSDL  
REST – requires a client that understand HTTP

# Forget Services, I want Events! - EDA

- **EDA** — “Event Driven Architecture is a software architecture pattern promoting the production, detection, consumption of, and reaction to events. An event can be defined as a significant change in state” - Wikipedia
- Services have historically been more request-response focused – synchronous processing – usually for responding to awaiting users.
- Events are more real-time alerting, no waiting – asynchronous processing – think push.
- Event-driven architecture can **complement** service-oriented architecture (SOA) because services can be activated by triggers fired on incoming events - Wikipedia.

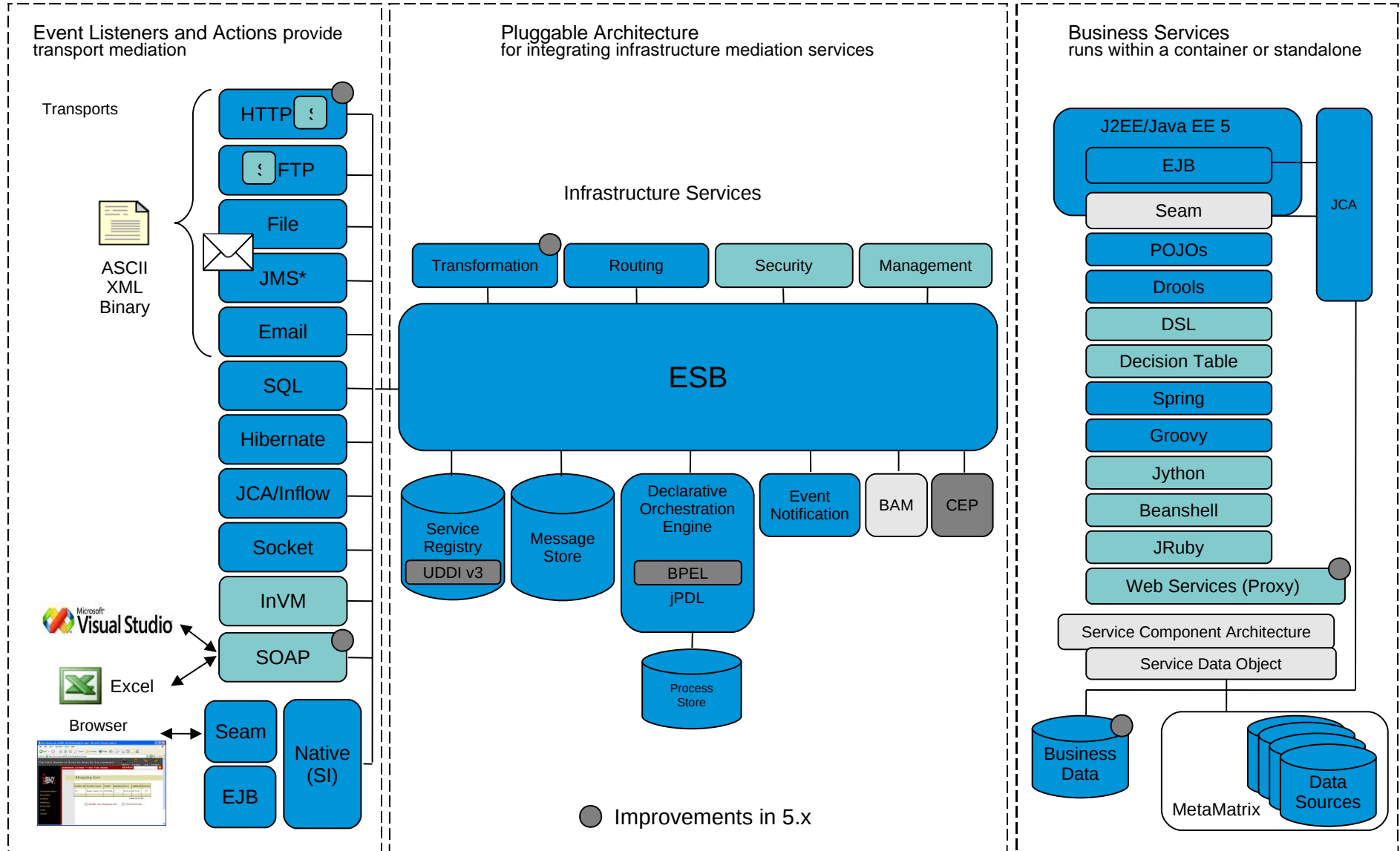
# Not Dead, just “resting” ;-)





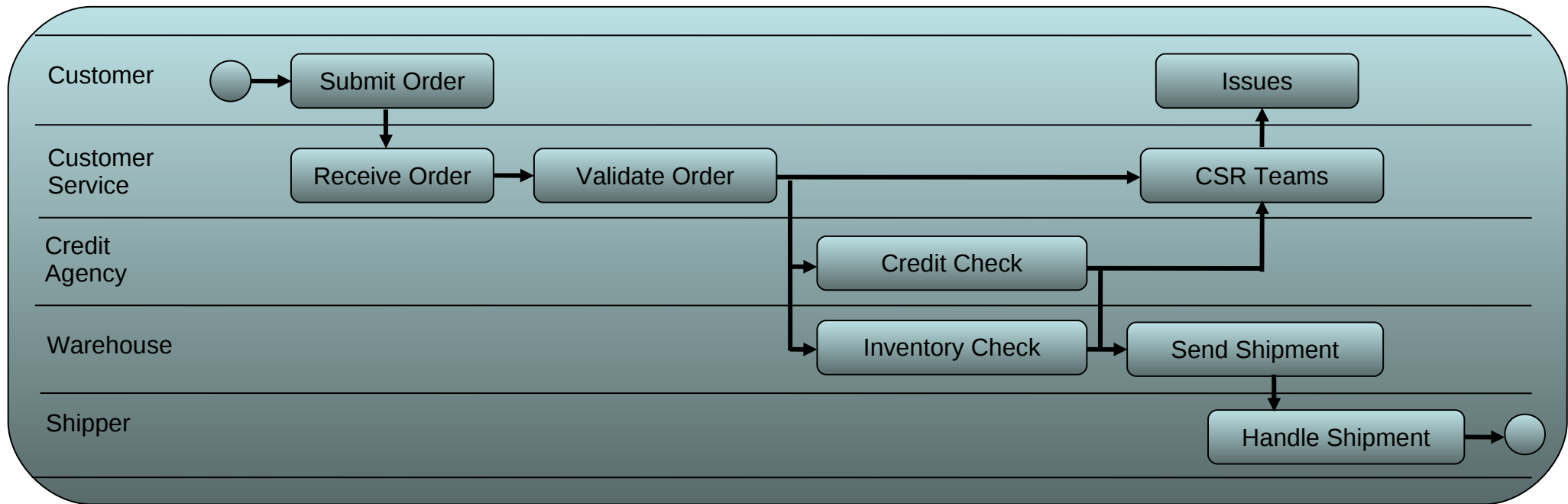
# JBoss SOA Middleware Overview





JMS\* - JBoss Messaging, IBM WebsphereMQ, TIBCO EMS, MRG-M

# Service Oriented Orchestration



## Validate Order

- a Parse XML
- b Transform
- c Apply Business Rules

## Credit Check

- a Create Outbound Msg
- b Handle Response
- c Apply Business Rules

## Send Shipment

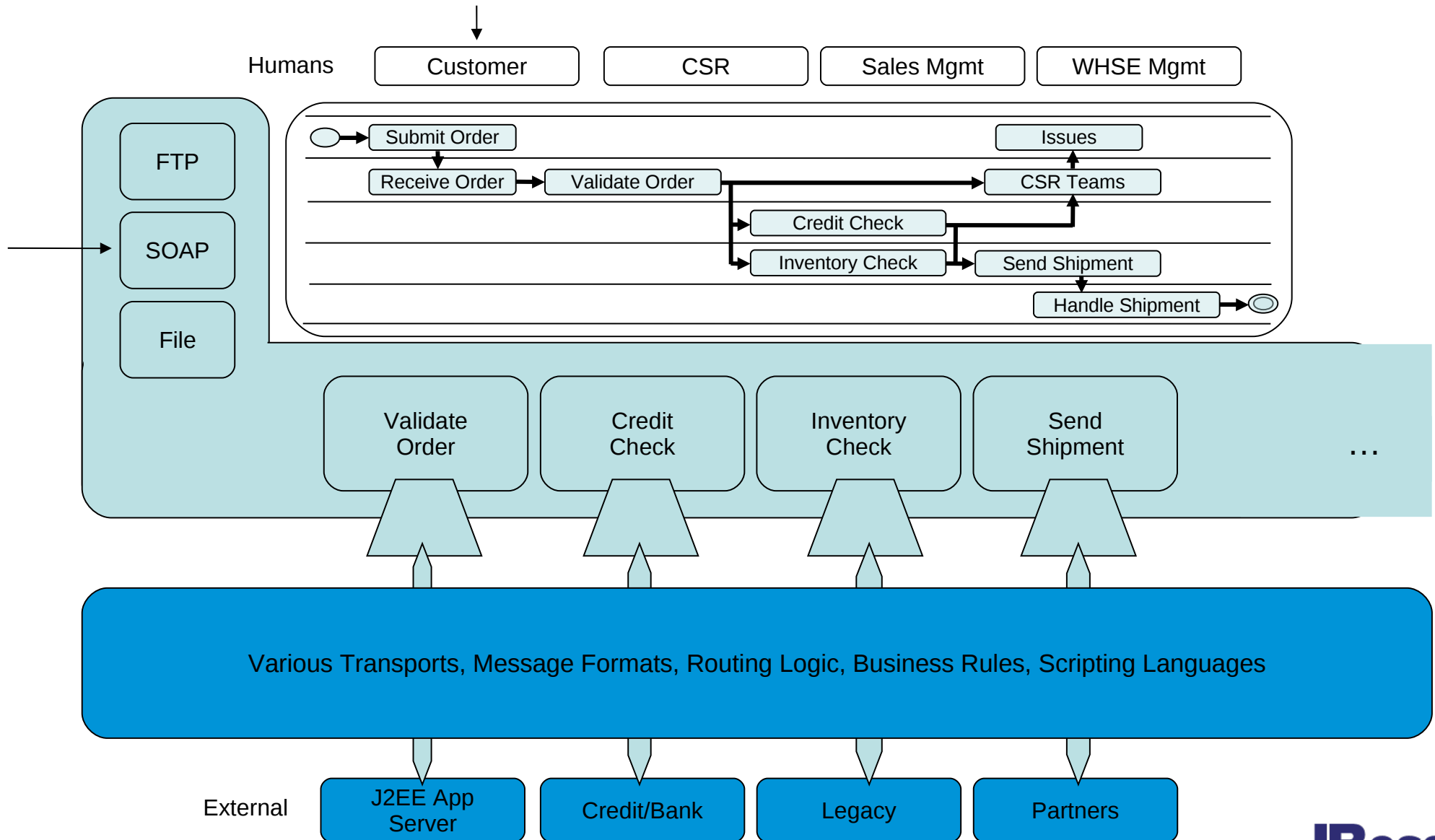
- a Determine Shipper(s)
- b Print Labels
- c Print Pick Tickets
- d Create & Send ASNs

ESB Mediates  
& Provides Services

## Inventory Check

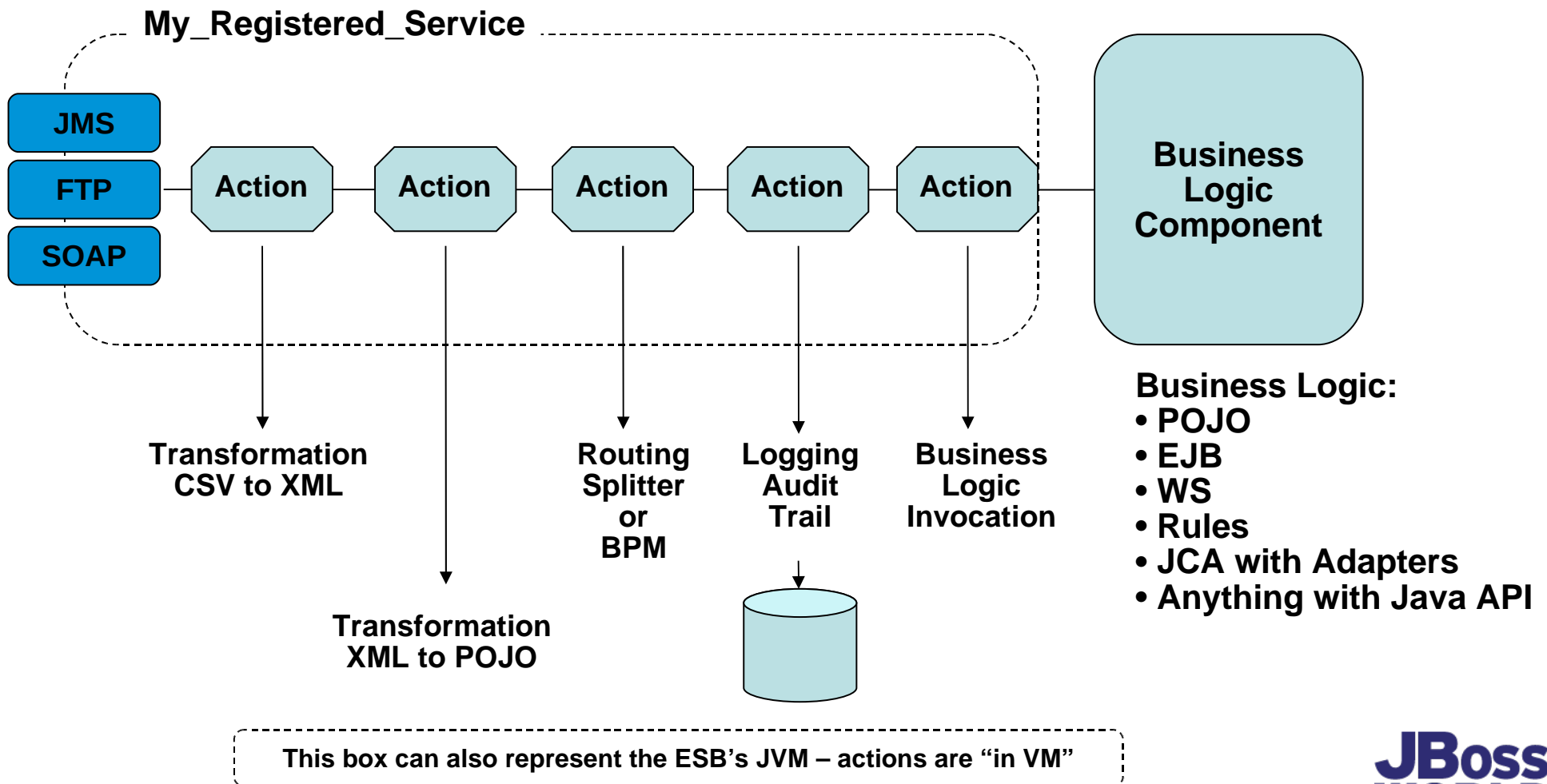
- a Send to N Warehouses
- b Handle N Responses
- c Determine Best WHSEs
- d Handle Drop-Ships

# Synergy of ESB + BPM + Rules

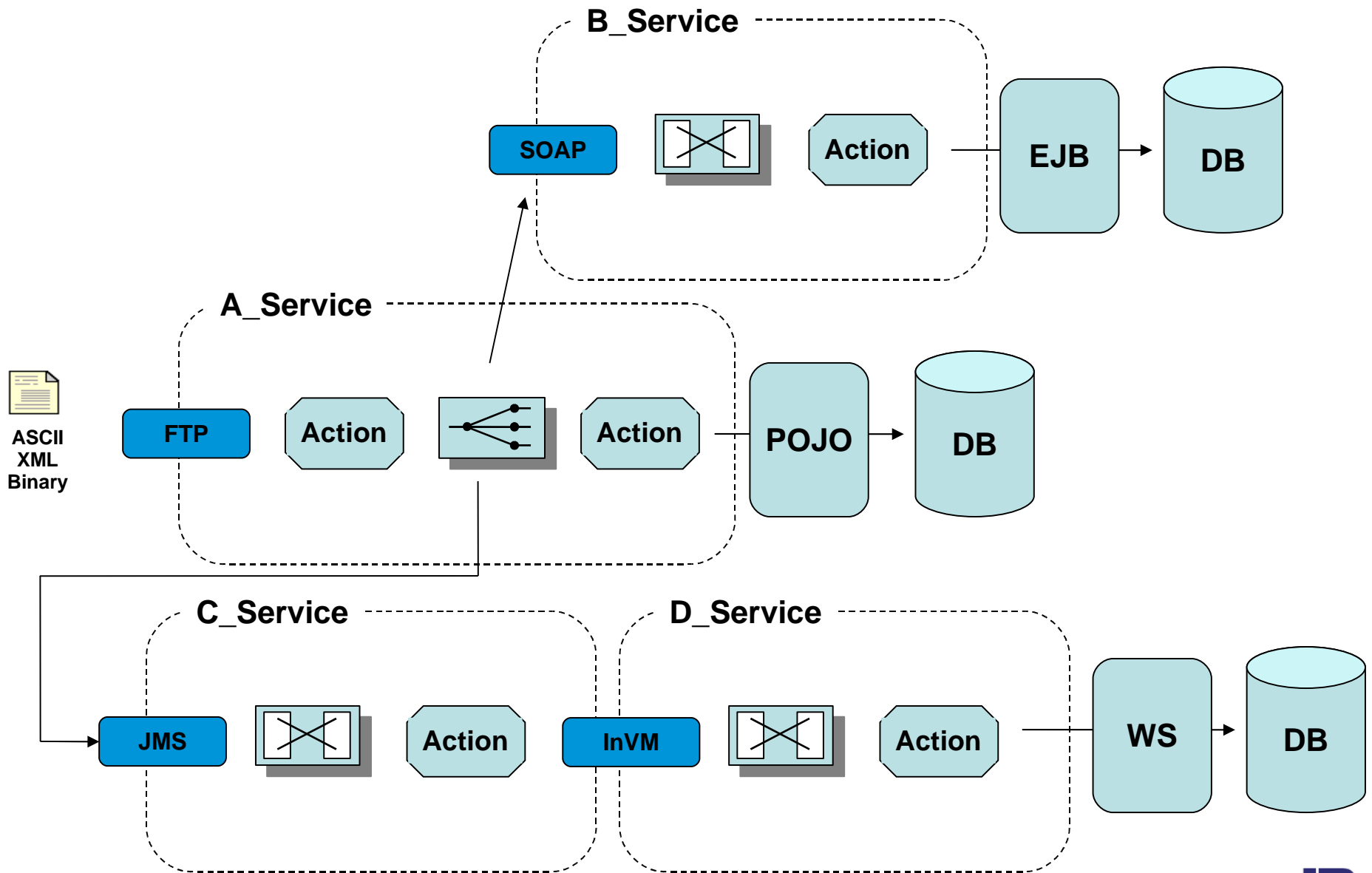


# The Action Chain

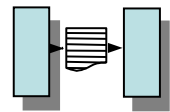
Actions are reusable mediation components that can be chained together to form the capabilities of a registered service. Actions can be dynamically added/removed at runtime.



# Services & Actions



# Based on Patterns



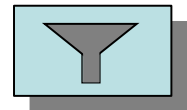
File Transfer



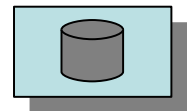
Channel



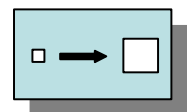
Message



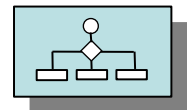
Message Filter



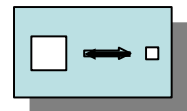
Message Store



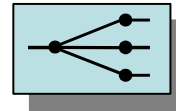
Enricher



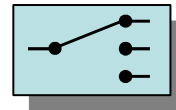
Process Manager



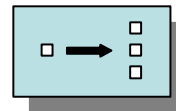
Content Filter



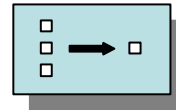
Recipient List



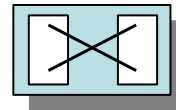
Router/CBR



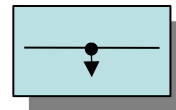
Splitter



Aggregator

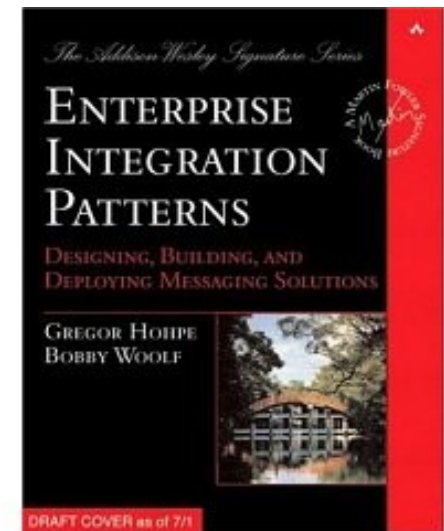


Translator/Transformer

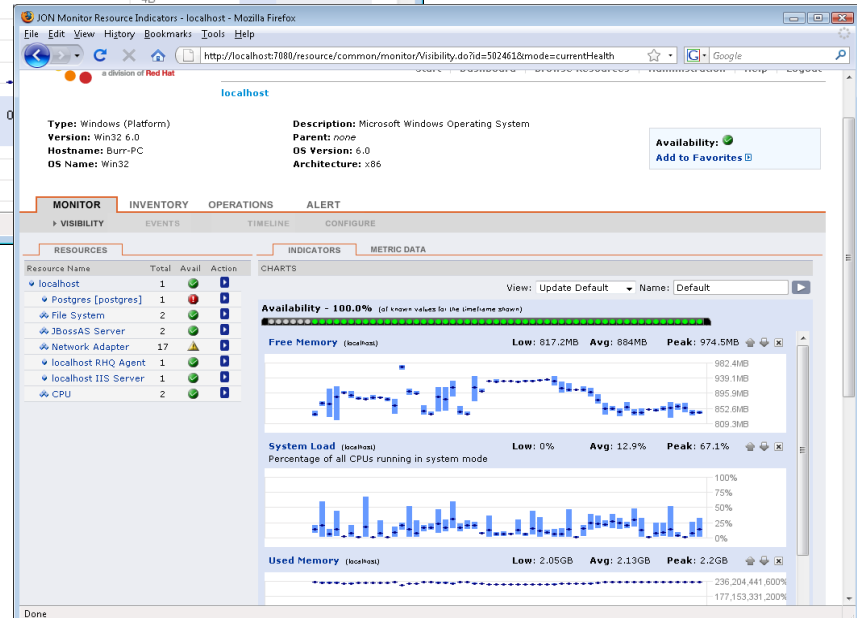
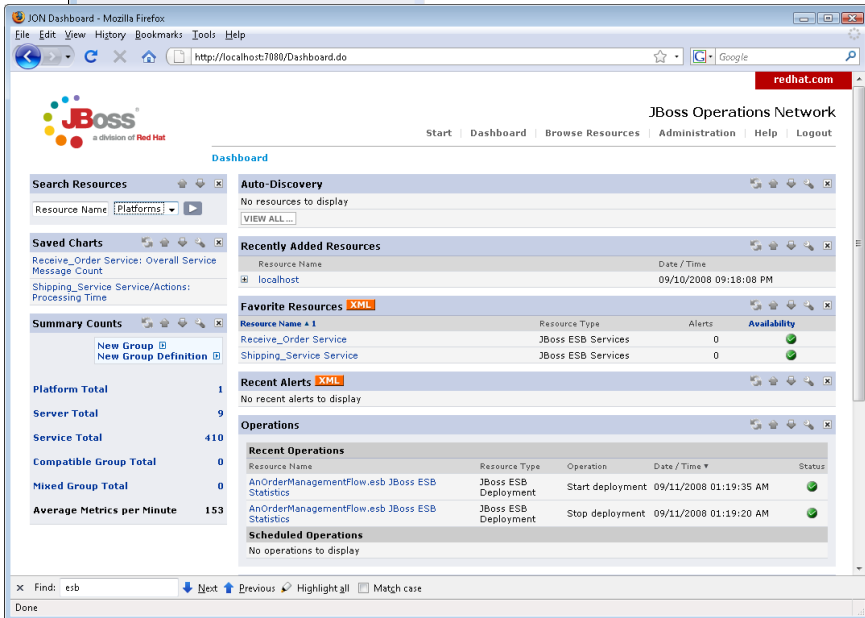
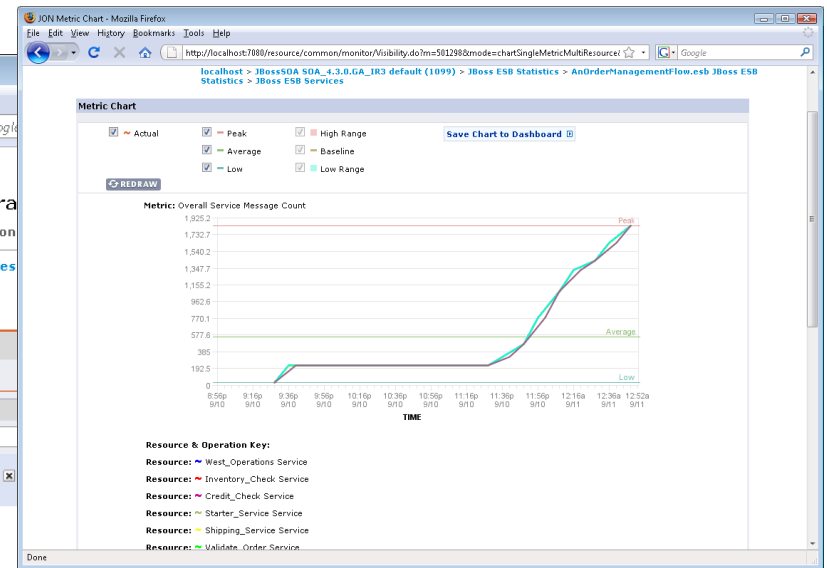
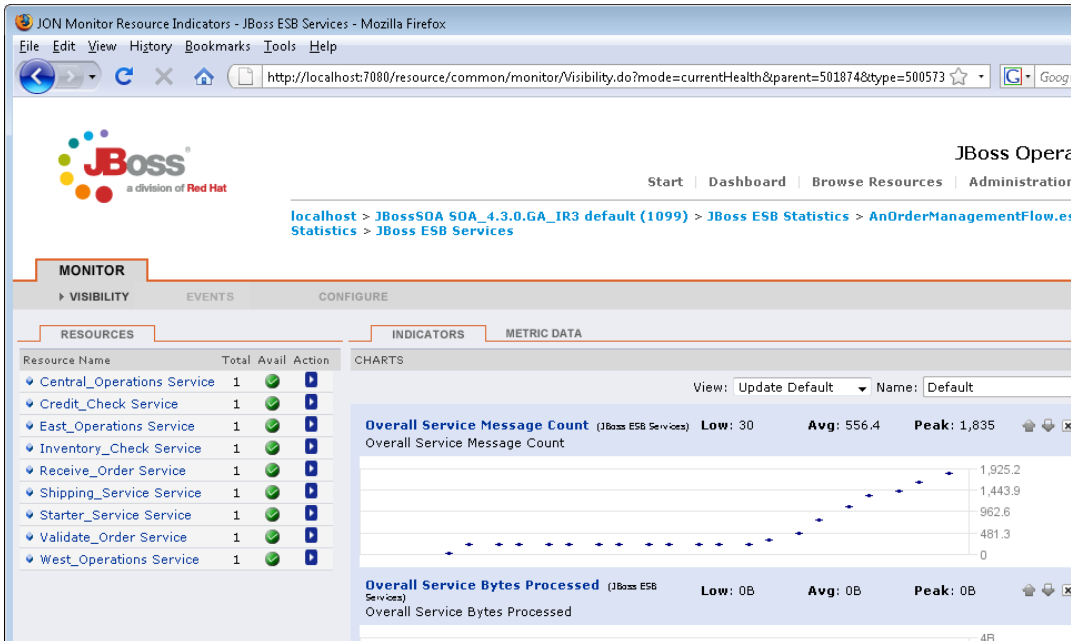


Wire Tap

More information at  
[www.enterpriseintegrationpatterns.com](http://www.enterpriseintegrationpatterns.com)



# SOA Monitoring and Management





**JBoss**  
**WORLD**  
CHICAGO 2009

**REST**



# Value of REST with an ESB

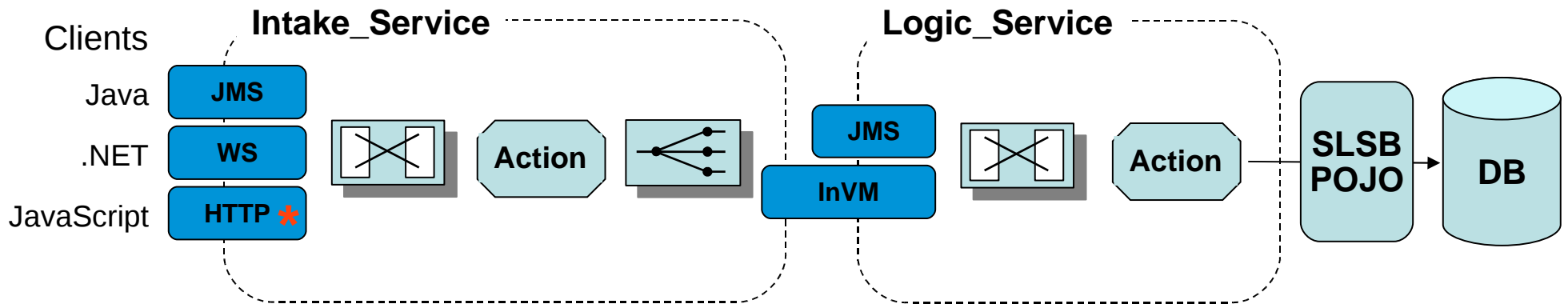
- JBossESB is not hard-wired into Web Services nor JMS, it is async event-driven and also allows for sync request-response.
- RESTful style interactions (GET, PUT, POST, DELETE) allow for Groovy, JavaScript, Ruby, etc clients to more easily consume a service.
- Now allows for different content types and use of HTTP response codes.
- REST is based on HTTP – new http-provider (ESB 4.7)

# New HTTP Provider (jboss-esb.xml)

```
<providers>
  <http-provider name="http">
    <http-bus busid="http_gateway"/>
    <exception mappingsFile="/http-exception-mappings.properties" />
  </http-provider>
</providers>

<services>
  <service category="Sales" name="List" description="" invmScope="GLOBAL">
    <listeners>
      <!-- Receives:http://<host>:<port>/MyESBArchive/http/sales/* -->
      <http-gateway name="sales"
        busidref="http_gateway"
        urlPattern="sales/*" />
    </listeners>
    <actions mep="RequestResponse">
      <action name="createAtomFeed" class="atom_publisher.MyAction"/>
    </actions>
  </service>
</services>
```

# ESB Transport Mediation

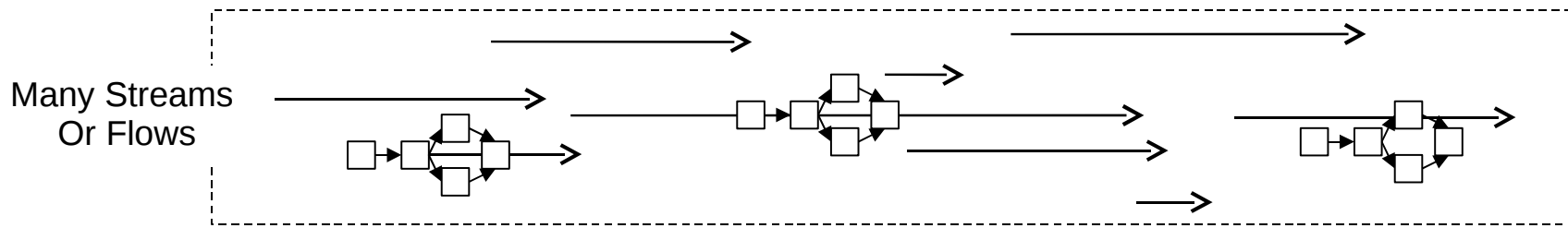




# ATOM Demo

# Complex Event Processing

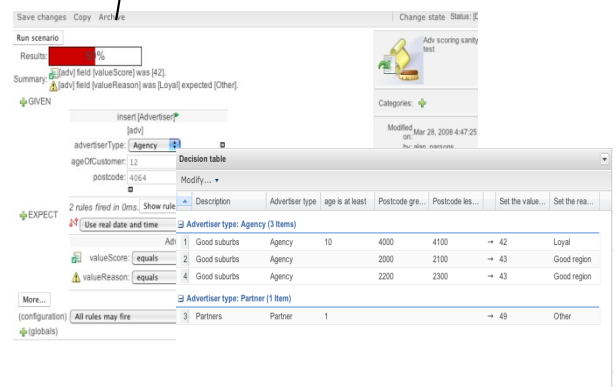


# ESP + CEP via ESB, Rules & JON

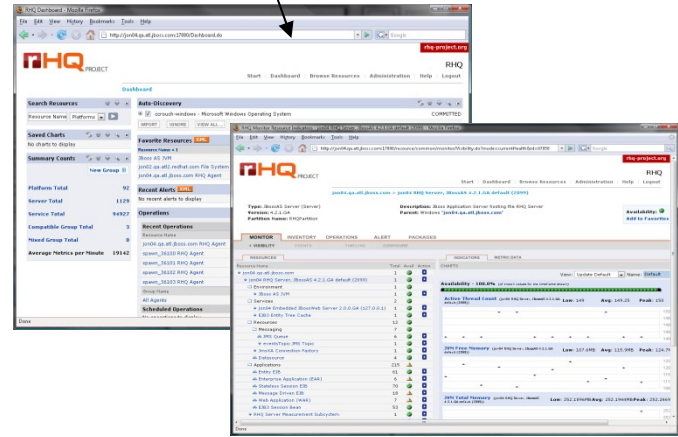


 **ESB: consumption, capture, transformation, routing, orchestration**  
**Rules: selection, aggregation, correlation, generation and publication** 

**Governance Tools**



**Repository: for editing, versioning, testing and publishing new rules and SOA artifacts**



**JON: start/stop services, monitor and alert on service and action-level performance, monitor and alert on business metric values**

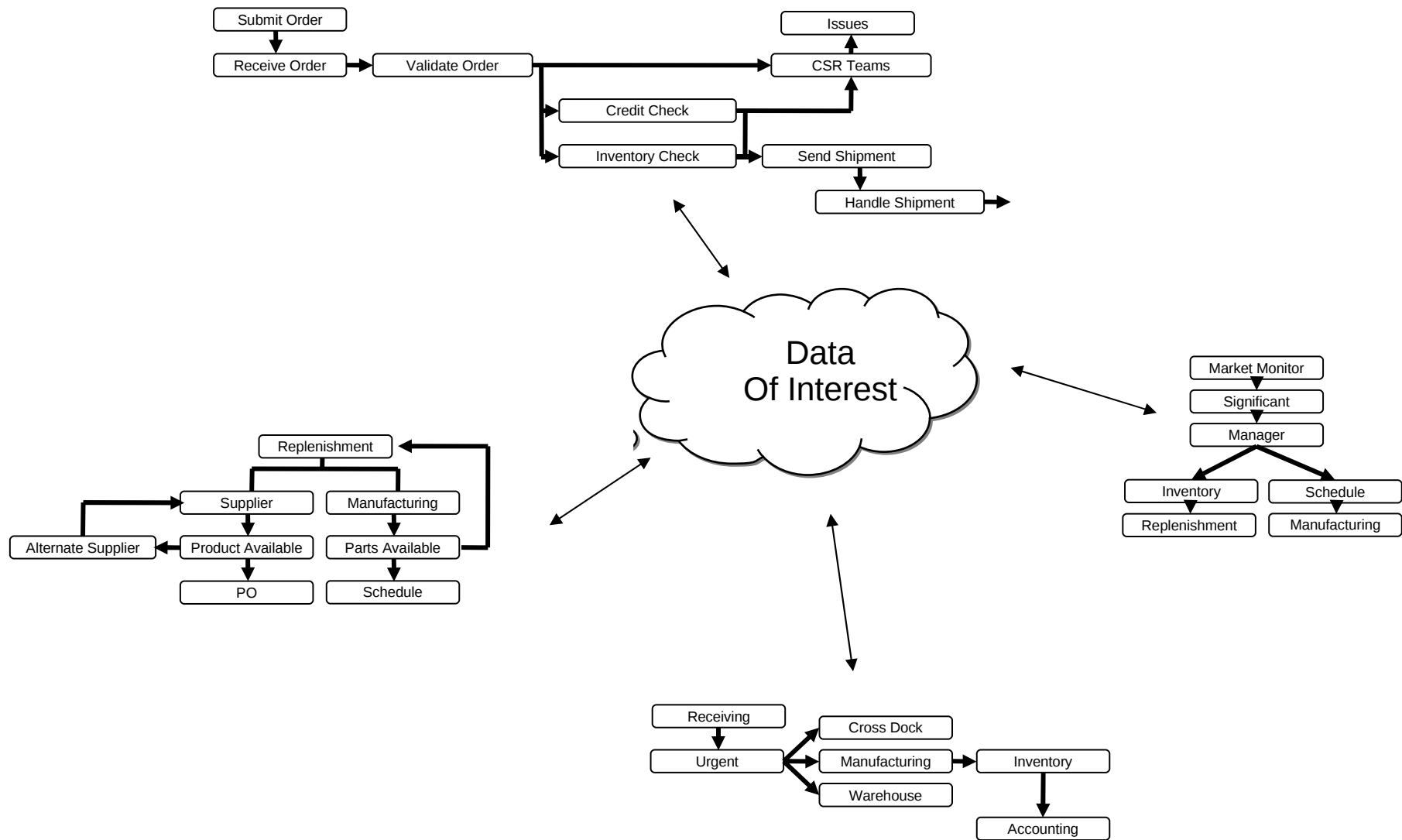
# Event Declaration & Selection

```
declare StockTick
    @role( event )
    @expires( 2m )
end

rule "average over last minute"
when
    $stat : Statistics( $symbol : symbol )
    Number( $av : doubleValue ) from accumulate(
        StockTick( symbol == $symbol, $p : price )
        over window:time( 1m )
        from entry-point "StockTick stream",
        average( $p ) )
then
    modify( $stat ) {
        average = $av
    }
end
```



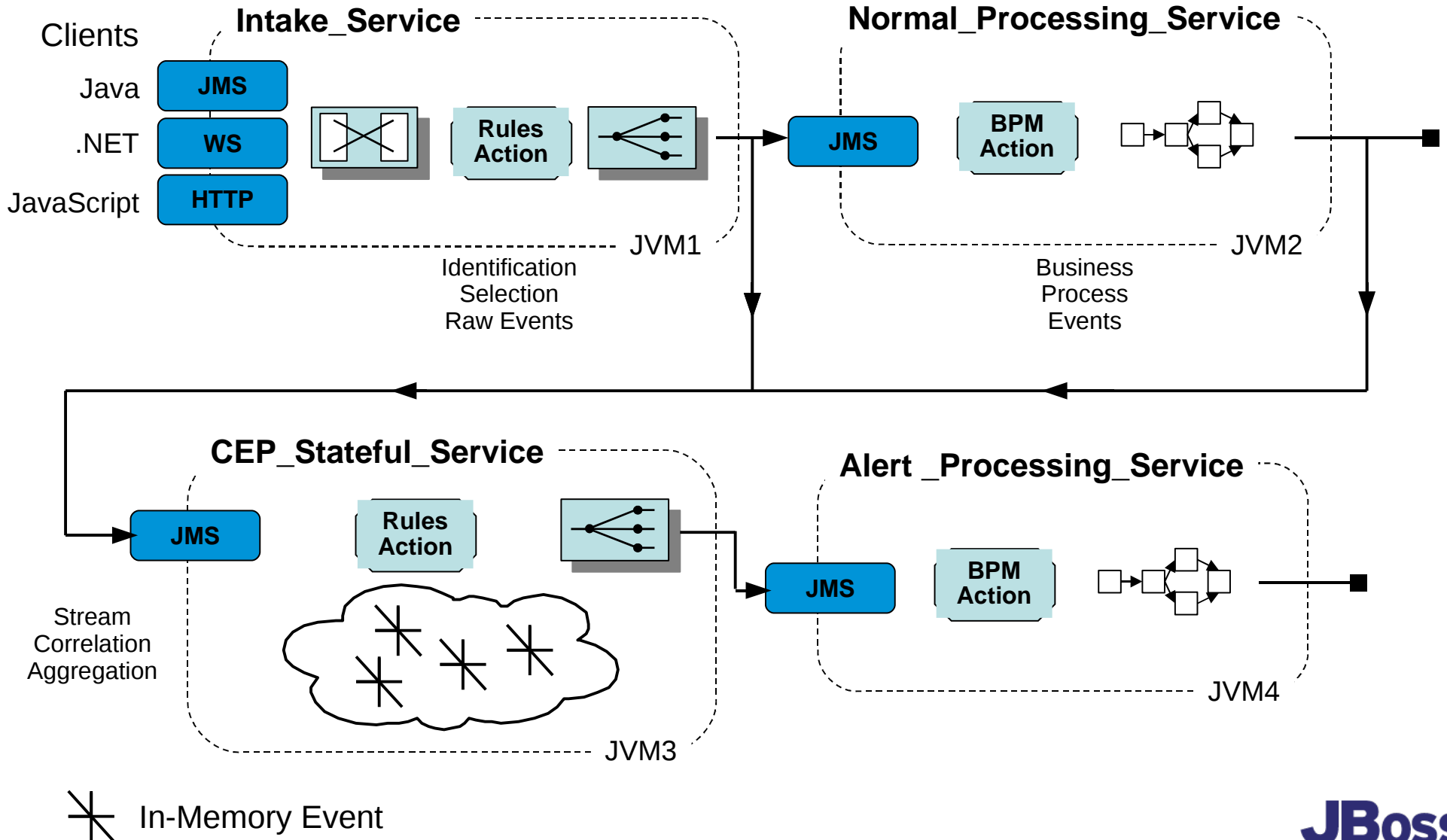
# Deal flow – Business Interceptor



# Value of CEP – Drools Fusion

- Stateful Rules Engines live in ESB Nodes, they retain the real-time history of the event flow
- Rules look for significant/interesting events in the stream, expiring older, insignificant events (memory management) – stateless rules engines/ESB nodes can route to stateful engines.
- Event data can be aggregated/accumulated over a time window or event series
- New events or messages back through the ESB can be generated, providing an alerting mechanism.

# CEP via ESB



# CEP Demo

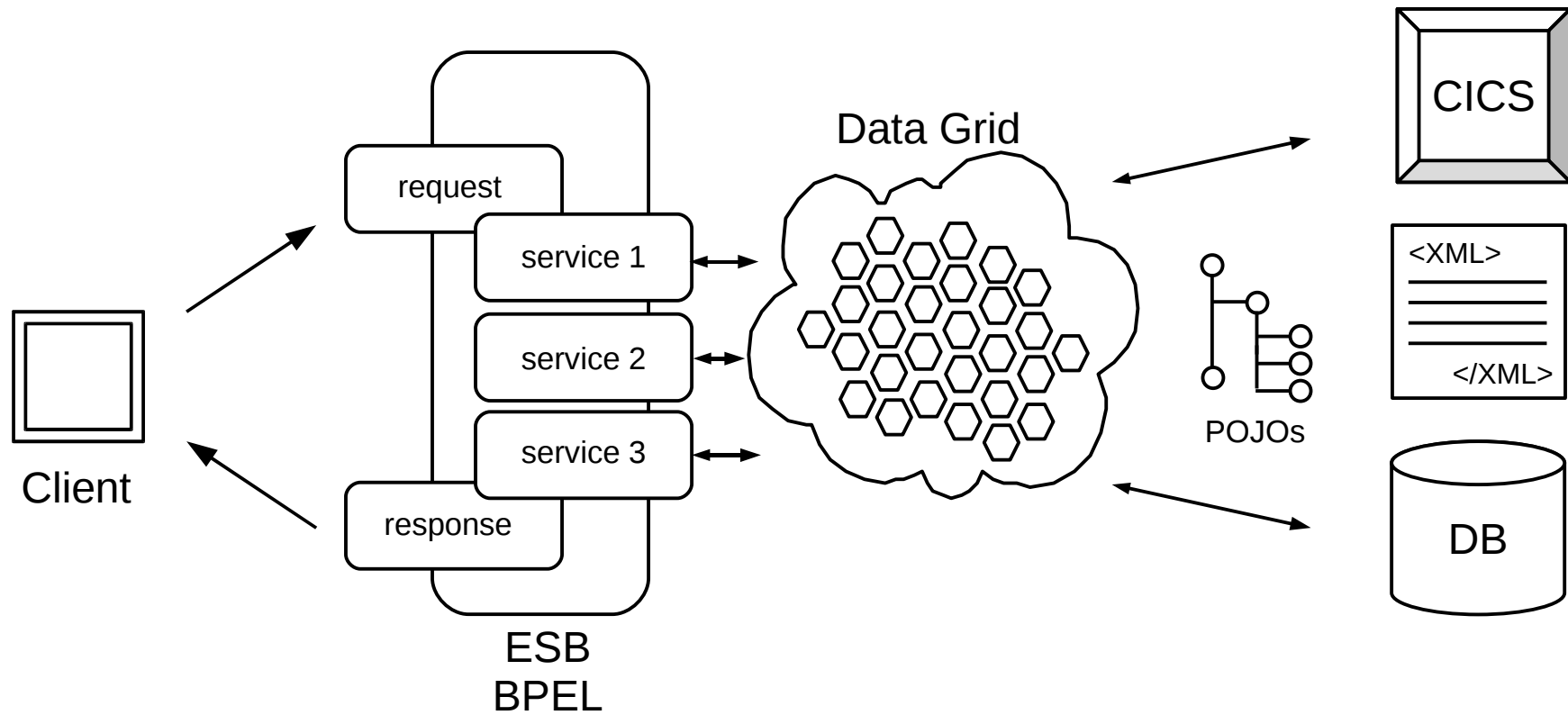
# Infinispan + ESB



# Quick Introduction: Infinispan

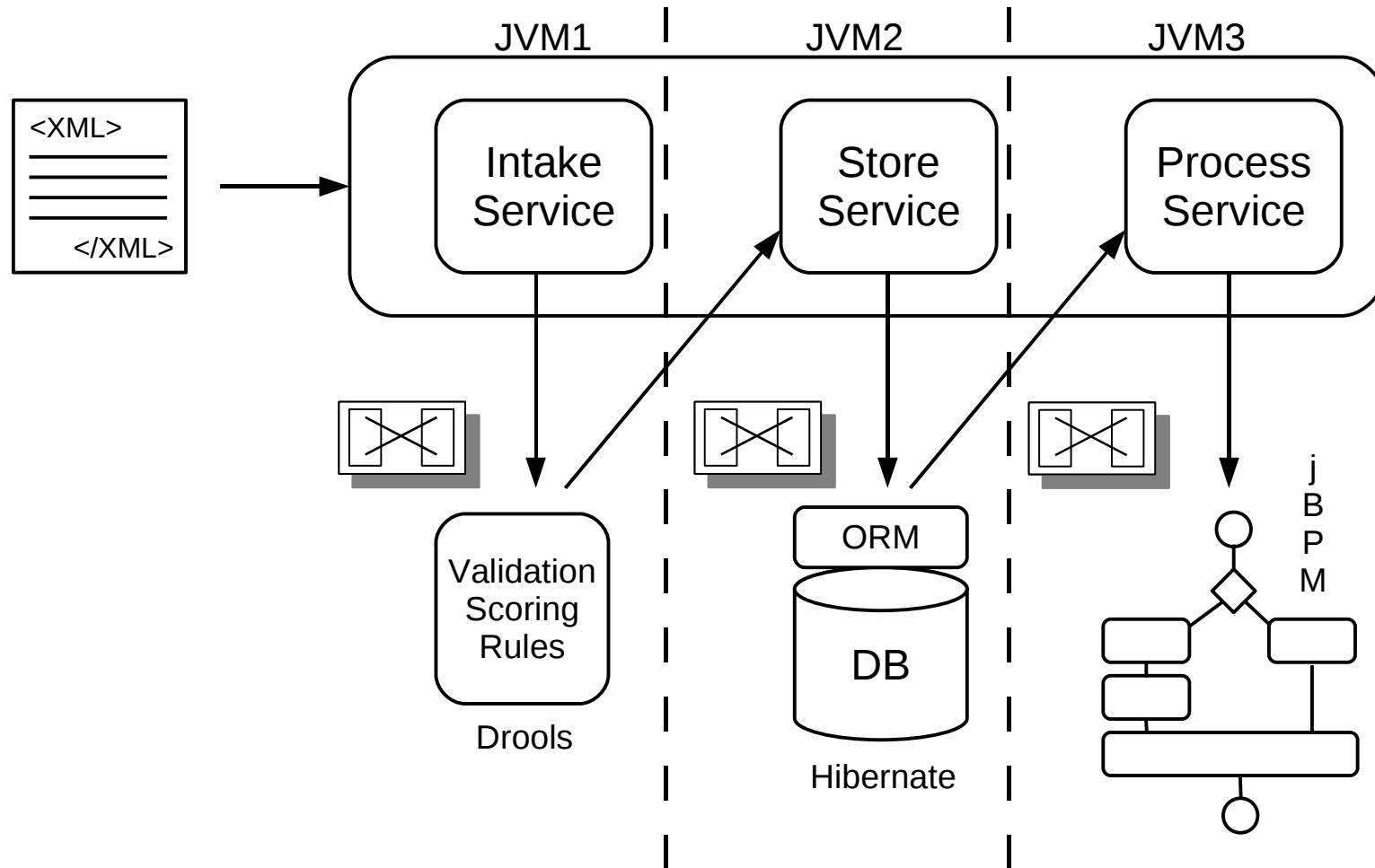
- Next Generation of JBoss Cache > Data Grid
- Still a “peer to peer” architecture
- JSR 107 – JCACHE
- Improve response times for service requests
- Reduce load on critical backend systems
- Improve fault tolerance, throughput, performance
- Linear scalability
- Predictable latency under increasing load
- Access to more than a single JVM's Heap

# Service Result Cache



Results of service invocations to “expensive” resources can be cached by the middle-tier architecture. Transformations to POJOs can be time consuming

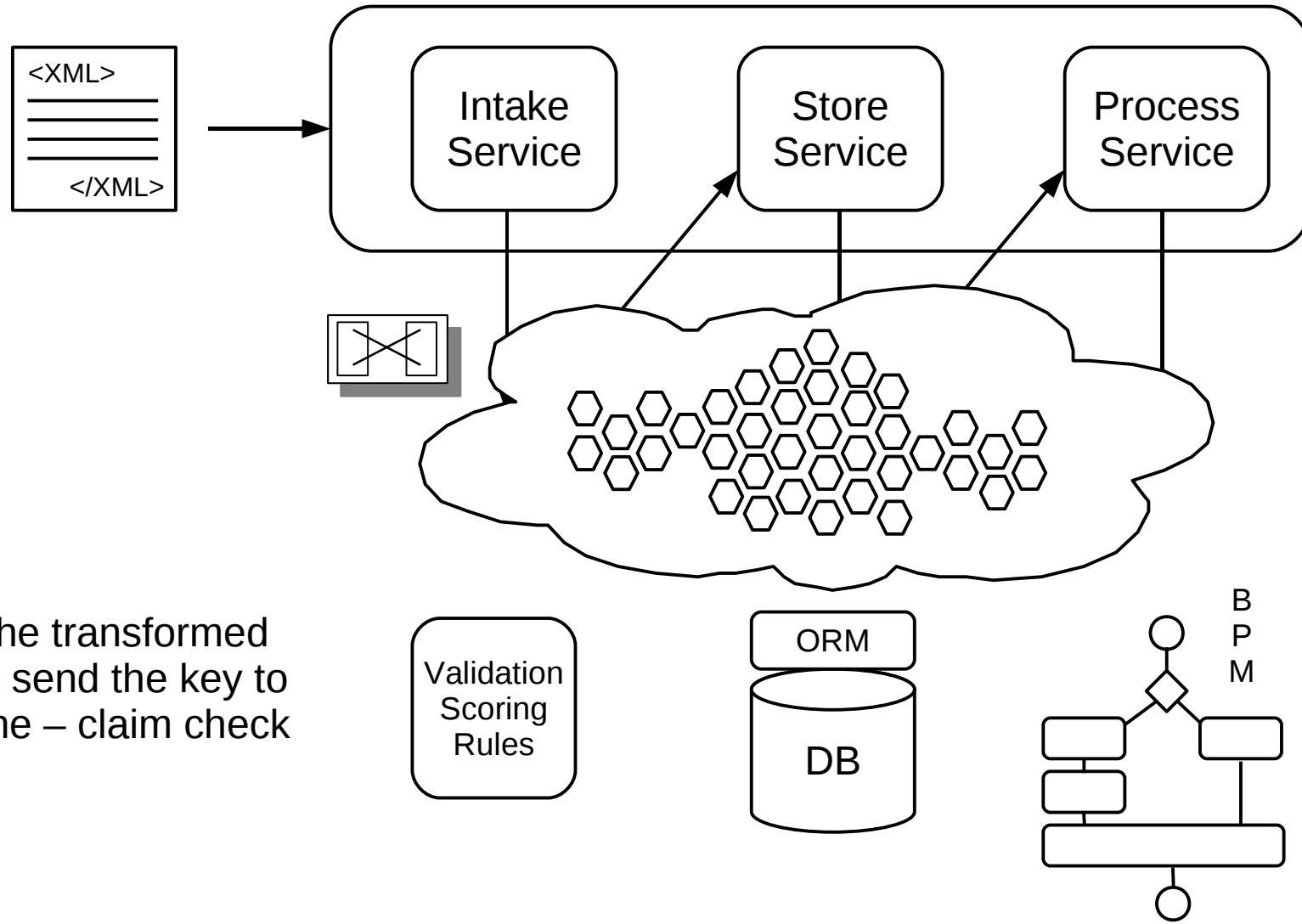
# Hoppity, Hop, Hop



Serializing large XML, transforming multiple times,  
serializing large POJOs can be expensive



# Still Hopping, less load



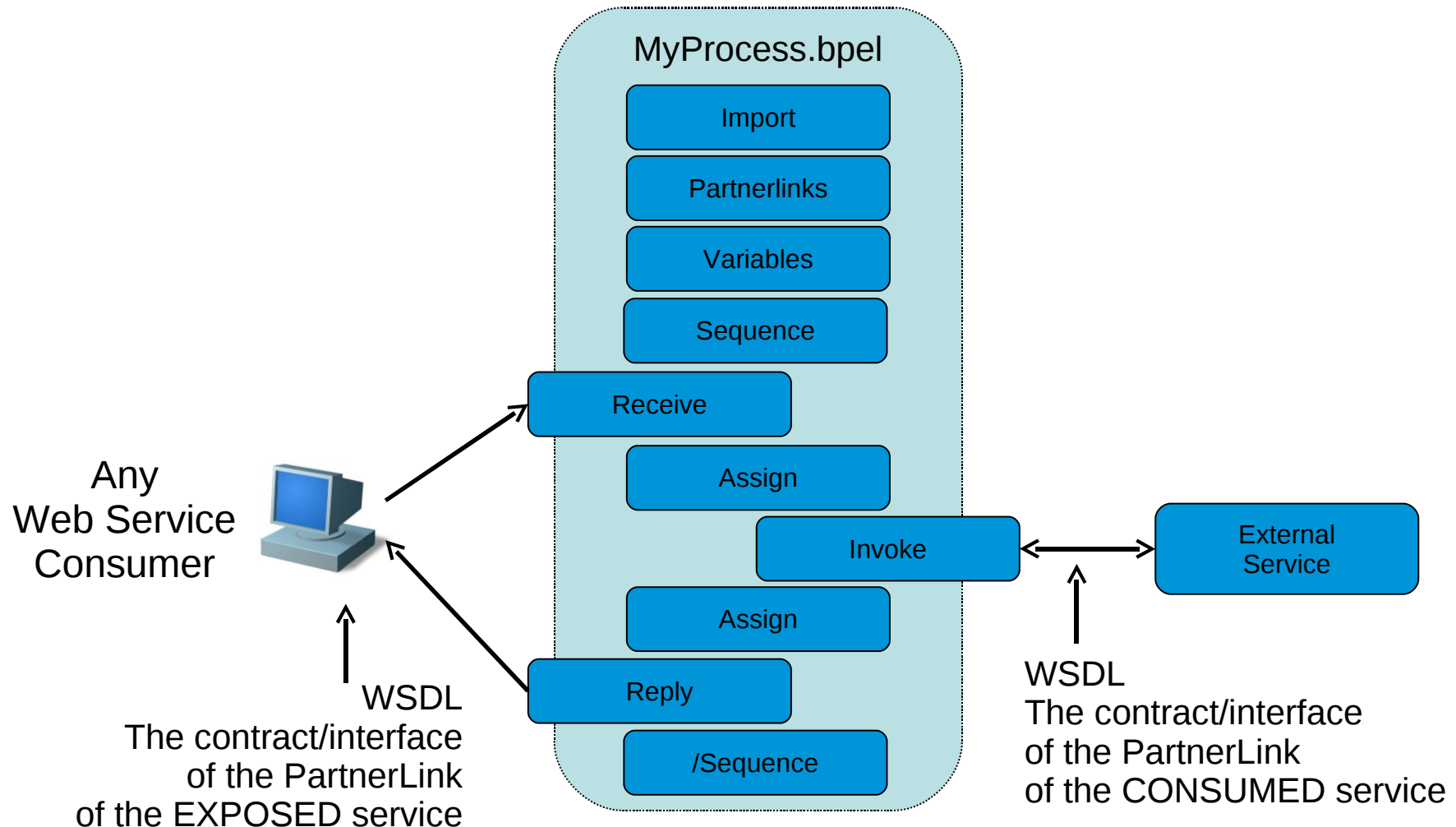
Keep the transformed version, send the key to the cache – claim check

# Data Grid Demo

# BPEL



# Simple BPEL Process Structure



# Tools – BPEL Editor

The screenshot displays the BPEL Editor interface for a process named 'OrderProcess.bpel'. The main workspace shows a flowchart with two sequence blocks. The first sequence block contains 'ReceiveCustomerOrder', 'AssignDataForNormalReply', and 'AcknowledgeReceiptOfOrder'. The second sequence block contains 'AssignDataForSalesOrder', 'CreateSalesOrder', 'WaitForNotificationFromOrderManager', 'AssignDataForOrderManagerAcknowledgement', 'AcknowledgeOrderManager', 'AssignDataForShipOrder', 'ShipOrder', and 'AssignDataForOrderConfirmation'. A 'Palette' on the right lists various BPEL activities and control structures. The 'Outline' window on the far right shows the process structure. At the bottom, the 'Properties' window for the 'ReceiveCustomerOrder' activity is open, showing 'Partner Link: Customer' and 'Operation: SubmitOrder'. A 'Quick Pick' list is also visible, containing 'Customer', 'RetailerPortType', 'SubmitOrder', 'OrderManager', 'RetailerCallbackPortType', and 'SendSalesOrderNotification'.

# Tools – WSDL Editor

The screenshot displays the WSDL Editor interface. At the top, there are tabs for 'OrderProcess.bpel', 'BPELRetailer.wsdl', 'Customer.wsdl', and 'OrderManager.wsdl'. The main workspace shows a service diagram with three components:

- OrderManagerPortTypeService**: Contains **OrderManagerPortTypePort** with URL `http://localhost:8865`.
- RetailerCallbackService**: Contains **RetailerCallbackSoap** with URL `http://localhost:8080/b...`.
- OrderManagerPortType**: Contains two operations:
  - cancelOrder**: Input: parameters; Output: result; Response: cancelOrderResponse.
  - customerOrder**: Input: parameters; Output: result; Response: customerOrderResponse. It also has a fault: SalesOrderFault.
- RetailerCallbackPortType**: Contains one operation:
  - SendSalesOrderNotification**: Input: Document; Output: Document; Response: salesOrderNotificationAck.

The **Outline** panel on the right shows a tree view of the WSDL structure, including Imports, Types, Services, Bindings, Port Type, and Messages.

At the bottom, the **Messages** panel is active, showing the following properties for **OrderManagerPortTypeService**:

General	Name:	OrderManagerPortTypeService
Documentation	Prefix:	tns
Extensions	Target namespace:	http://org.jboss.esb/quickstarts/bpel/ABI_OrderManager

An **Advanced...** button is located at the bottom right of the Messages panel.

# BPEL Resources

- <http://www.jboss.org/riftsaw>
- Riftsaw Forums
- WS-BPEL Primer:  
<http://docs.oasis-open.org/wsbpel/2.0/Primer/wsbpel-v2.0-Primer.pdf>
- WS-BPEL 2.0 Specification:  
<http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>
- Apache Ode  
<http://ode.apache.org/>

# Professional Enterprise Support

- RESTful HTTP – SOA Platform 5 (Q1 2010)
- CEP – SOA Platform 5 (Technology Preview)
- Infinispan – SOA Platform 5.1 or 6.0 (TBD)
- BPEL – BPEL Platform 2.0 (Q1 2010)

## For more information at JBoss World 2009

- REST & HTTP on ESB – Kevin Conner (Newcastle)
- CEP via Drools Fusion – Edson Tirelli (Toronto)
- Infinispan – Manik Surtani (London)
- BPEL – John Graham (Boston)



**QUESTIONS?**

**TELL US WHAT YOU THINK:  
[REDHAT.COM/JBOSSWORLD-SURVEY](https://redhat.com/jboss-world-survey)**

**FOLLOW US:**

TWITTER.COM/REDHATSUMMIT

**TWEET ABOUT US:**

ADD #SUMMIT AND/OR #JBOSSWORLD TO THE END  
OF YOUR EVENT-RELATED TWEET

