SOA for Developers

The Architect's and Coder's View of the Service Oriented Universe

Speaker

- Burr Sutter – burr.sutter@jboss.com
- Sun Java Champion
- President – Atlanta Java Users Group (www.ajug.org) – Top 50 JUG
- Chapter President – International Association of Software Architects, Atlanta (www.iasahome.org)
- Formerly Chief Technologist at BravePoint, responsible for Java EE and .NET consulting practices

Agenda

- SOA Defined
- SOA Approaches
- Focus on Architecture
- Stages of SOA Adoption
- Quick Takeaways
- REST vs Web Services
- Web Services via JBossWS
- SOAP & WSDL
- WS-*
- SOA & Web 2.0
- BPEL via JBoss jBPM
- JBI – Java Business Integration

MPAA Rating

Demos

- JSR 181 Annotated Web Services (easiest process)
- VB.NET WinForm and C# WebForm Consumers
- EJB Web Service Endpoint (Seam endpoint: DVDStore)
- SOAP Sniffer – TcpMon/SoapUI
- 181 Web Service that returns an array of POJOs (complex types)
- 181 Web Service using JBoss Rules
- SOAP Header Handlers*
- JBossWS AJAX Consumer
- BPEL w/ .NET Consumer
- WS-Addressing*
- WS-Security Interop*
- WS-Security & JAAS*
- WS-Eventing*
- MTOM* (not yet available on JBossWS 1.0, in HEAD)
- MS Office 2003/2007 Integration*
- Starting from WSDL and XSD*
- SOA Composite Application Platform*

* Under Development

SOA Defined

SOA Defined - OMG

In April 2006 The Object Management Group's (OMG) SOA Special Interest Group adopted the following definition for SOA:

Service Oriented Architecture *is an architectural style for a community of providers and consumers of services to achieve mutual value*, that:

- Allows participants in the communities to work together with **minimal co-dependence or technology dependence**
- Specifies the contracts to which organizations, people and technologies must adhere in order to participate in the community
- Provides for business value and business processes to be realized by the community
- Allows for a variety of technologies to be used to facilitate interactions within the community

Source: Wikipedia

SOA Defined - OASIS

In March 2006 the OASIS group SOA Reference Model released its first public review draft. This defines the basic principles of SOA that apply at all levels of a service architecture, from business vision through to technical and infrastructure implementation.

Service-Oriented Architecture; *A paradigm for organizing and utilizing distributed capabilities that **may be under the control of different ownership domains**. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations.*

Source: Wikipedia

Principles of SOA

- Services share a formal contract
- Services are loosely coupled
- Services abstract underlying logic
- Services are composable
- Services are reusable
- Services are autonomous
- Services are stateless
- Services are discoverable

Source: Thomas Erl - SearchWebService.com

SOA Approaches

SOA Approaches

- Focus on N-Tier development with stateless(ful), UI-less, SQL-less, middle-tier business and technically focused "services" which can be easily modified and redeployed.
- SOA is simply the next OO (but I thought AOP was the next OO)
- CORBA, DCOM, Jini, RMI
- Web Services
  - ✓ Interoperable
  - ✓ Multi-Vendor Support
  - ✓ Emerging Standards (OASIS, W3C, etc)
  - ✓ Real Application to Application (A2A, B2B) INTEGRATION

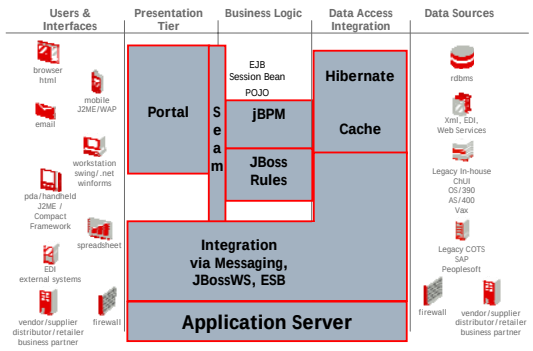
Stages of Adoption

Stages of Adoption

- Design/Determination
  - ✓ Should everything be a service?
- Enablement (Java EE, .NET, various)
- Infrastructure
- Orchestration/Composition
  - ✓ BPEL
  - ✓ Transformation
  - ✓ Routing
  - ✓ Integration: Data, Application, Information, Process
- Discoverable
- Service Governance
- Dynamic, Event-Driven, Intelligent Enterprise

Focus on Architecture

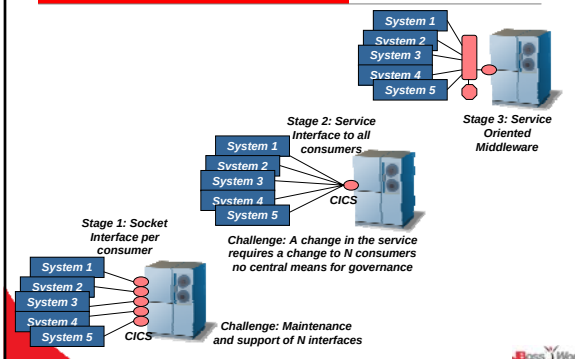
JEMS & Java EE Architecture



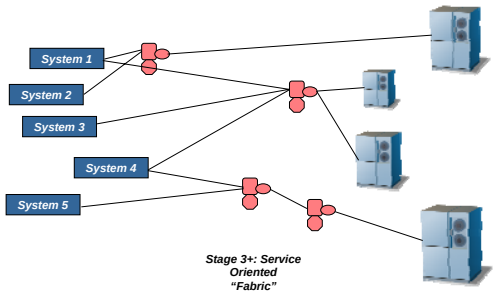
Swivel Chair Integration

Modern organizations are using FTP and custom socket code

Evolutionary Transition

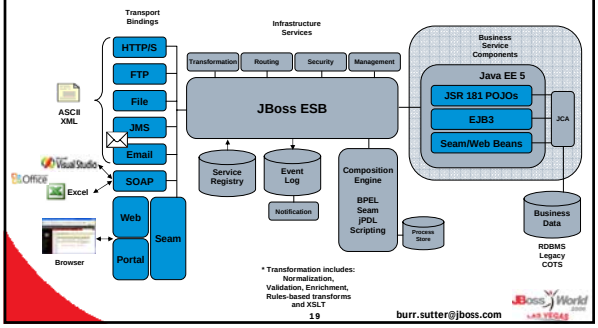


Distributed Intermediaries



Enterprise Service Bus

- Enterprise Service Bus is a "narrowing" of the concepts in SOA. A concrete implementation of SOA principles.



Quick Takeaways

A few tidbits to walk away with…

Quick Takeaways

- Enterprise (big biz) SOA is about integration
- Every IT shop has embarked on an SOA implementation – your CIO says so
- The magic number is 40, real number is under 10 of deployed Web Services
- N-Tier development of coarse-grained, loosely-grained "services"
- Reuse is NOT the goal
- Streamlined processes around core services that are easily modified thus enabling a more agile and response IT infrastructure is the single greatest value
- The task of designing reusable services, the discovery, learning and team building created by the exercise is valuable all by itself
- Read: Enterprise Integration Patterns and Enterprise Service Bus (Chappell)

REST vs SOAP

The REST of the Story

- Representational State Transfer from Roy Fielding's doctoral thesis
- The architectural style of the Web
  - ✓ Everything is a resource
  - ✓ Every resource is addressable by URI
  - ✓ Operations on resources defined by HTTP verbs
- RESTafarians: don't like the fact that SOAP makes everything a POST

REST-based Services

- http://hostname/customer/abc123

	SQL	HTTP
Create	INSERT	PUT
Read	SELECT	GET
Update	UPDATE	POST
Delete	DELETE	DELETE

REST vs SOAP

SOAP/WS-*	REST
700+ pages of Spec	1 Page
Tools & Vendors	RESTLets
WSDL	MS Word Doc
UDDI	Google
Proxy Generation	Unnecessary
Too Granular/RPC	Exposes all URIs
	Web Caching

JBossWS 1.0

Included in JBoss App Server 4.0.4.GA w/EJB3 profile (using installer)

JBossWS 1.0 Standard Features

- RPC style endpoints
- Document style endpoints (wrapped and unwrapped)
- SOAP header values bound/unbound to endpoint parameters
- J2EE endpoint deployment model for EJB and Java
- J2EE client deployment model
- Dynamic Invocation Interface (DII)
- JAXRPC client/server side handlers
- Holders for INOUT/OUT parameters

JBossWS 1.0 Extended Features

- Message style endpoints
- Attachments Profile Version 1.0
- Dynamic client/server side handler injection
- Support for WS-Security
- Support for WS-Addressing and JSR-261 (JAX-WSA)
- Standard Web Service Metadata as annotations (JSR-181)
- XML binding delegated to the JBossXB project
- Transport delegated to the JBoss Remoting project
- Support for EJB-3.0 endpoints

JBossWS 1.0 Non-Interop

- According to various forum postings…
- Microsoft SOAPToolkit 3.0, typically used by VB 5/6, VBScript (ASP) and VBA (Excel macros) seems to be incompatible with JBossWS 1.0.
- It adds a name space to the parameters/request names, it defaults to rpc/encoded.
- gSoap library sometimes used by C++ developers seems to have the same problem. Try to use Doc instead of RPC.
- Perl Lite seems to also have a similar problem.
- RPC Encoded is DEAD!
- JBossWS is less forgiving than the previous Axis-based stack.

JSR-181 Annotations

- @WebService
  - ✓ marks a Java class as implementing a Web Service
- @WebMethod
  - ✓ identifies the individual methods
- @SOAPBinding
  - ✓ makes the service available over SOAP 1.1
- @WebParam
  - ✓ customizes mapping of an individual parameter
- @WebReturn
  - ✓ customizes mapping of the return value

The 181 Recipe (like Servlet Easy)

1. Create an exploded WAR in the 4.0.4.GA w/EJB3 "deploy" directory.
2. Create sub-directories: WEB-INF\classes\org\jboss\samples
3. Create the annotated WS called HelloWorldWS.java:

```
package org.jboss.samples;

import javax.jws.WebMethod;
import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;
import javax.jws.soap.SOAPBinding.Use;


@WebService(name = "HelloWorld",
      targetNamespace="http://com.burrsutter.jbossws/helloworld")
@SOAPBinding(style = SOAPBinding.Style.RPC)
public class HelloWorldWS
{
   @WebMethod
   public String sayHello(String toWhom)
   {
       System.out.println("I'm Hit! " + toWhom);
      return "Hello World: " + toWhom + " on " + new java.util.Date();
   }
}
```


The 181 Recipe (like Servlet Easy) Part II

4. Compile using Java 5 with this CLASSPATH:

```
set JBOSS_HOME=C:\JBoss\jboss-4.0.4.GA
set CLASSPATH=.;%JBOSS_HOME%\server\default\deploy\jbossws.sar\jbossws.jar
```

5. Create a web.xml file in the WEB-INF directory with the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation= "http://java.sun.com/xml/ns/j2ee
     http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
     version="2.4">
<servlet>
<servlet-name>HelloWorldService</servlet-name>
<servlet-class>org.jboss.samples.HelloWorldWS</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>HelloWorldService</servlet-name>
<url-pattern>/HelloWorldService</url-pattern>
</servlet-mapping>
</web-app>
```


The 181 Recipe (like Servlet Easy) Part III

6. Using your browser hit: http://localhost:8080/jbossws and click on View
7. You should see a listing with http://yourhost:8080/MyWS/HelloWorldService?wsdl where "MyWS" was the name of the exploded WAR directory created in Step 1. Click on the link to view the WSDL
8. For .NET Interop, download and install Microsoft Visual Basic Express Edition. Start a new Windows Application project. Add a Web Reference to the URL listed in step 7.

```
Dim proxy As New JBossHelloWorld.HelloWorldWSService
Dim result As String
result = proxy.sayHello(TextBox1.Text)
MsgBox(result)
' where JBossHelloWorld was the name given while adding the Web
    Reference
```


JBoss World 2006 LAS VEGAS

Demo

My 181 Process

181 Tips & Tricks

Tip: In the @WebMethod annotation for the sayHello method add (operationName="SayHello") as this will produce proxy.SayHello() on the VB side. A .NET person expects to have their method/function names begin with a capital letter.

Trick: JBossWS 1.0 copies your web.xml to web.xml.org and creates its own web.xml. This web.xml fails to redeploy when the AppServer boots up or when "touched", simply delete and rename web.xml.org back to web.xml.

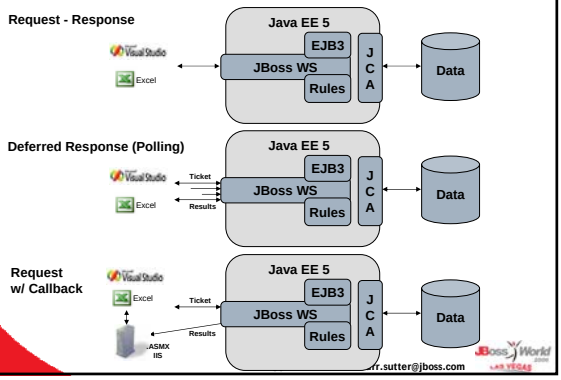
Trick: Collections not currently supported, use Arrays

Trick: Really complex objects not currently supported

Tip: This recipe doesn't provide you with the Java client-side which would be useful for automated unit testing. When you have the basic working, build an Ant script similar to the ones in the JBossWS samples.

Tip: Do consider the use of NAnt and NUnit for automated building/integration and unit testing on the .NET side.

WS Architecture



AJAX Consumer

- Download and use: http://www.ibm.com/developerworks/webservices/library/ws-wsajax/
- Create Document/Literal WebServices for this IBM toolkit:

```
@SOAPBinding(style = SOAPBinding.Style.DOCUMENT)
```


AJAX Consumer Code

```
<html><head>
<script type="text/javascript" src="ajaxws/prototype.js"></script>
<script type="text/javascript" src="ajaxws/ws.js"></script>
<script type="text/javascript">
function callServer(myName) {
  alert("before: " + myName);
  var wsCall = new WS.Call('/MyWS/HelloWorldDocument');
  var rpcFunction = new
      WS.QName('sayHello','http://com.burrsutter.jbossws/helloworld');
  wsCall.invoke_rpc(rpcFunction, new Array({name:'String_1',value:myName}),null,
      function(wsCall, envelope)
      {
        alert("callback");
        var soapBody = envelope.get_body();
        var helloResponse = soapBody.get_all_children()[0];
        var helloReturn = helloResponse.get_all_children()[0];
        $('displayarea').innerHTML = helloReturn.get_value();
        $('soap').innerHTML = arguments[2].escapeHTML();
      } // callback function
  );
}
function $(elementid) {
  return document.getElementById(elementid);
}
</script>
</head><body><form>
  Name: <input type="text" id="myName"/>
  <input type="button" value="Click Me" onClick="callServer($('myName').value)" />
</form>
<hr/><div id="displayarea"></div> <hr/> <div id="soap"></div> </body> </html>
```

Demo

AJAX Consumer

Introduction to SOAP

```
POST /MyWS/HelloWorldService HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client
    Protocol 2.0.50727.42)
Content-Type: text/xml; charset=utf-8
SOAPAction: ""
Host: 127.0.0.1:9090
Content-Length: 350
Expect: 100-continue
Connection: Keep-Alive

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Header/>
<soap:Body>
    <sayHello xmlns="http://com.burrsutter.jbossws/helloworld">
    <String_1 xmlns="">Say Your Name</String_1>
    </sayHello>
</soap:Body>
</soap:Envelope>
```


Introduction to WSDL

- How does it map to my method signatures?
- Complex Types (xs:import vs definitions)
- RPC/Encoded (Dead)
- RPC/Literal (.NET 2.0)
- Document/Literal (WS-I BP 1.0)
- http://www.w3.org/TR/2001/NOTE-wsdl-20010315
- Tweak the WSDL from name="result" to name="parameters" in JBossWS 1.0 http://jira.jboss.com/jira/browse/JBWS-771
- Interop Notes: http://wiki.jboss.org/wiki/Wiki.jsp?page=JBossWSAndDotNet

Introduction to WSDL

- <defintions>
- <types>
- <messages>
- <portType><operation>
- <binding>
- <service>

Hello World 181 WS

```
package org.jboss.samples;

import javax.jws.WebMethod;
import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;
import javax.jws.soap.SOAPBinding.Use;


@WebService(name = "HelloWorld",
    targetNamespace="http://com.burrsutter.jbossws/helloworld")
@SOAPBinding(style = SOAPBinding.Style.RPC)
public class HelloWorldWS
{
   @WebMethod
   public String sayHello(String toWhom)
   {
      System.out.println("I'm Hit! " + toWhom);
      return "Hello World: " + toWhom + " on " +
        new java.util.Date();
   }
}
```


Hello World WSDL

```
<definitions name='HelloWorldWSService' targetNamespace='http://com.burrsutter.jbossws/helloworld'
      xmlns='http://schemas.xmlsoap.org/wsdl/' xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'
      xmlns:tns='http://com.burrsutter.jbossws/helloworld'
      xmlns:xsd='http://www.w3.org/2001/XMLSchema'>
 <types></types>
 <message name='HelloWorld_sayHelloResponse'>
  <part name='result' type='xsd:string'/>
 </message>
 <message name='HelloWorld_sayHello'>
  <part name='String_1' type='xsd:string'/>
 </message>
 <portType name='HelloWorld'>
  <operation name='sayHello' parameterOrder='String_1'><!-- methods -->
   <input message='tns:HelloWorld_sayHello'/><!-- see messages for input params types -->
   <output message='tns:HelloWorld_sayHelloResponse'/><!-- see messages for return type -->
  </operation>
 </portType>
 <binding name='HelloWorldBinding' type='tns:HelloWorld'>
  <soap:binding style='rpc' transport='http://schemas.xmlsoap.org/soap/http'/><!-- rpc over http -->
  <operation name='sayHello'>
   <soap:operation soapAction=''/>
   <input>
    <soap:body namespace='http://com.burrsutter.jbossws/helloworld' use='literal'/>
   </input>
   <output>
    <soap:body namespace='http://com.burrsutter.jbossws/helloworld' use='literal'/>
   </output>
  </operation>
 </binding>
 <service name='HelloWorldWSService'>
  <port binding='tns:HelloWorldBinding' name='HelloWorldPort'>
   <soap:address location='http://bsutter:9090/MyWS/HelloWorldService'/><!-- 9090 is TCPMON -->
  </port>
 </service>
</definitions>
```


Demo

Tcpmon

Arrays of POJOs

```
// Java WS Producer Code
public class POJOServiceDoc {
 @WebMethod (operationName="GetAllPeopleArray")
 public Person[] getPeople()
 { … }
```

```
public class Person {
  private String name;
  private int age;
  private Date birthDate;
  private Address address;
  public void setName (String name)
  { this.name = name; }
  public String getName()
  { return name; }
```

Java getters/setters become .NET properties
Address.java is not shown on this slide

```
' VB.NET WS Consumer Code
Dim proxy As New JBossPOJO.POJOServiceService
Dim myPeople As JBossPOJO.Person() = proxy.GetAllPeopleArray
For Each myPerson As JBossPOJO.Person In myPeople
    MsgBox(myPerson.name & " " & myPerson.address.streetAddr1)
Next
```

```
// C# WS Consumer COde
JBossPOJO.POJOServiceService proxy = new JBossPOJO.POJOServiceService();
JBossPOJO.Person[] myPeople = proxy.GetAllPeopleArray();
StringBuilder aString = new StringBuilder();
foreach (JBossPOJO.Person myPerson in myPeople)
{
  aString.Append(myPerson.name + " " + myPerson.address.streetAddr1 + "\n");
}
TextBox1.Text = aString.ToString();
```

With JBossWS 1.0, tweak the WSDL

POJOs in the WSDL

```
<complexType name='Address'>
    <sequence>
     <element name='city' nillable='true' type='string'/>
     <element name='country' nillable='true' type='string'/>
     <element name='postalCode' nillable='true' type='string'/>
     <element name='state' nillable='true' type='string'/>
     <element name='streetAddr1' nillable='true' type='string'/>
     <element name='streetAddr2' nillable='true' type='string'/>
    </sequence>
</complexType>
<complexType name='Person'>
    <sequence>
     <element name='address' nillable='true' type='tns:Address'/>
     <element name='age' type='int'/>
     <element name='birthDate' nillable='true' type='dateTime'/>
     <element name='name' nillable='true' type='string'/>
    </sequence>
</complexType>
```

Fix the generated WSDL

```
<message name='POJOServiceDoc_GetAllPeopleArrayResponse'>
  <part element='tns:GetAllPeopleArrayResponse' name='result'/> <!-- Fix 'result' -->
</message>

<message name='POJOServiceDoc_GetAllPeopleArrayResponse'>
 <part element='tns:GetAllPeopleArrayResponse' name='parameters'/> <!-- To 'parameters' -->
</message>
```


Demo

POJOs/Complex Types
Seam Bean + EJB3 + WS

WS JBoss Rules Recipe

- Follow the 181 Recipe
- Drop the various JAR files from JBoss Rules/Drools into the WAR's lib folder.
- Use the new Rules Designer to build a .DRL ruleset, debug and test.
- Drop the .DRL file into the WAR
- Build a 181 WS that executes the rule.
- Compile the new WS but add the Rules/Drools JARs into the CLASSPATH.
- Create your own Ant script to accomplish these tasks.

A Simple DRL

StockFlagger.drl uses Stock.java

```
package com.sample;

import com.sample.Stock;

rule "IdentifyHotBuys"
    no-loop true
    salience 10
when
  stock : Stock (
  indexName == "DJIA",
  currentPrice <= 100.0,
  technicalScore >= 10.1,
  fundamentals   >= 62.5,
  attractiveness >= 87.2)
then
  System.out.println("Meets the BUY conditions: " + stock);
  stock.setFlag(true);
  stock.setAction("BUY");
  modify( stock );
end
```

```
package com.sample;
public class Stock { // Stock.java
  private String symbol;
  private String indexName;
  private float currentPrice;
  private float technicalScore;
  private float fundamentals;
  private float attractiveness;
  private boolean flag = false;
  private String action="HOLD";

  public String getSymbol() {
    return this.symbol;
  }
  public void setSymbol(String symbol) {
    this.symbol = symbol;
  }
  public String getIndexName() {
    return this.indexName;
  }
  public void setIndexName(String indexName) {
    this.indexName = indexName;
  }
  public float getCurrentPrice() {
    return this.currentPrice;
  }
  public void setCurrentPrice(float currentPrice) {
    this.currentPrice = currentPrice;
  }
  . . .
```


Rules wielding Web Service

```
@WebService(name = "StockFlagger",
    targetNamespace="http://com.burrsutter.jbossws/stockflagger")
@SOAPBinding(style = SOAPBinding.Style.DOCUMENT)
public class StockFlaggerWS
{
   @WebMethod(operationName="CheckStocks")
   public Stock[] checkStocks(Stock[] stocks)
   {
      WorkingMemory wm = acquireWorkingMemory();
      for (Stock stock : stocks) { // load into working memory
        wm.assertObject(stock);
      }
      wm.fireAllRules();
      return stocks;
   }
   private WorkingMemory acquireWorkingMemory()
   {
     WorkingMemory wm = null;
     try {
       Reader drlFile = new
        InputStreamReader (
       Thread.currentThread().getContextClassLoader().
         getResourceAsStream("com/sample/StockFlagger.drl"));
       PackageBuilder builder = new PackageBuilder();
       builder.addPackageFromDrl(drlFile);
       Package pkg = builder.getPackage();
       RuleBase ruleBase = RuleBaseFactory.newRuleBase();
       ruleBase.addPackage(pkg);
       wm = ruleBase.newWorkingMemory();
     } catch (Exception e) { System.out.println(e); }
     return wm;
   }
} // StockFlaggerWS
```

Demo

JBoss Rules + JBossWS

WS-*

WS-*

Q2 2006

- **JBossWS 1.0**
  - ✓ SOAP 1.1
  - ✓ WSDL 1.1
  - ✓ UDDI 2.0 & access via JAXR 1.0
  - ✓ WS4EE 1.1
  - ✓ JAXRPC 1.1
  - ✓ SAAJ 1.2
  - ✓ XML 1.0
  - ✓ XML Schema Part 1: Structures
  - ✓ XML Schema Part 2: Data types
  - ✓ WS-I BP 1.0
  - ✓ WS-Security 1.0
  - ✓ JSR-261 JAX-WSA
  - ✓ WS-Eventing (moving to WS-Notification)
  - ✓ WS-Addressing 1.0
- **JBoss Transactions 4.2**
  - ✓ WS-Context
  - ✓ WS-Coordination 1.0
  - ✓ WS-AtomicTransaction 1.0
  - ✓ WS-BusinessActivity 1.0
  - ✓ WS-Addressing 1.0 (OASIS Interop) as well as August 2004

Q3 2006

- **JBossWS 1.1**
  - ✓ JAX-WS 2.0
  - ✓ SOAP 1.2
  - ✓ MTOM
  - ✓ WS-Policy
- **jBPM**
  - ✓ WS-BPEL 1.0 & 2.0 (Public Draft)

Awaiting Prioritization/Partners

- ✓ WS-Federation (Passive)
- ✓ WS-Security 1.1
- ✓ SAML Token Profile 1.0
- ✓ WS-Trust
- ✓ WS-Metadata Exchange
- ✓ WS-Secure Conversation
- ✓ WS-Discovery
- ✓ WS-Reliable Messaging/Exchange
- ✓ WS-Management
- ✓ WS-Transfer
- ✓ WS-Enumeration

Use Jira Voting!!!

JavaEE 5 & WS-* http://wiki.jboss.org/wiki/Wiki.jsp?page=JBossWSSpecStatus

Specification/Standard	% Complete	Java EE 5	JBossWS Version
WS-I Basic Profile 1.0	100%	Yes	1.0.0
WS-I Basic Profile 1.1	100%	Yes	1.0.0
Attachments Profile 1.0	100%	Yes	1.0.0
MTOM	30%	Yes	1.0.0
OASIS XML Catalog	90%	Yes	
JAX-RPC 1.1	100%	Yes	1.0.0
JSR-109	100%	Yes	1.0.0
JSR-109MR	70%	Yes	
JAXB 2.0	20%	Yes	
JAX-WS 2.0	10%	Yes	
JSR-181	100%	Yes	1.0.0
JSR-181MR2	90%	Yes	
SAAJ 1.2	100%	Yes	1.0.0
SAAJ 1.3	80%	Yes	
JSR-261	100%	No	1.0.0
WS-Security 1.0	99%	No	1.0.0
WS-Policy	100%	No	1.0.0
WS-Eventing	100%	No	1.0.0
WS-Addressing	100%	No	1.0.0

WS-* (JBoss AS 4.0.4 via JBossWS 1.0)

- WS-Eventing
  - ✓ Provides a protocol that allows web services to subscribe to/register for event notification messages. Sink=receiver/consumer & Source=sender/producer
- WS-Security
  - ✓ Provides SOAP header extensions for client authentication (username/password, x509), message integrity (XML Signature) and message confidentiality (XML Encryption).
- WS-Addressing (JSR 261)
  - ✓ Provides SOAP header extensions to specify:
    Message Destination – wsa:To
    Source Endpoint – wsa:From
    Reply Endpoint - wsa:ReplyTo
    Fault Endpoint – wsa:FaultTo
    Action – wsa:Action
    Unique Message ID
    Relationship to other messages
    Parameters

WS-Security

- Describes an interoperable way to authenticate the message sender, ensure message integrity, encrypt messages and prevent replay attacks.
- SSL can provide encryption from a consumer to a producer in a point-to-point fashion but what if you don't want intermediaries reviewing the content, especially if those intermediaries log the messages.
- WS-I UserName Token Profile 1.0
- WS-I X.509 Certificate Profile 1.0
- WS-I Basic Security Profile 1.0

Before WS-Security

```
public String login(String userName, String password) {
   // check for valid login
   return securityToken;
}
public CustomerOrder getOrder(String securityToken,
   String orderID) {
   // first validate the securityToken
   // then fetch the order
   return foundCustomerOrder;
}
public int saveNewOrder(String securityToken,
   CustomerOrder newOrder) {
   // again, validate the securityToken
   // then validate and process the order
}
```

Authentication was programmatically handled. State was held in the HttpSession and encryption was via SSL thus locking you into the HTTP transport.

Before WS-Addressing

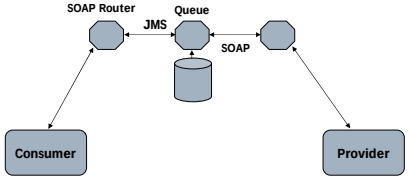
```
POST /MyWS/HelloWorldService HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services
    Client Protocol 2.0.50727.42)
Content-Type: text/xml; charset=utf-8
SOAPAction: ""
Host: 127.0.0.1:9090
Content-Length: 357
Expect: 100-continue
Connection: Keep-Alive

<?xml version="1.0" encoding="utf-8"?>
   <soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <soap:Body>
         <sayHello xmlns="http://com.burrsutter.jbossws/helloworld">
            <String_1 xmlns="">Marc Fluery</String_1>
         </sayHello>
      </soap:Body>
   </soap:Envelope>
```

A SOAP request has a transport-based address

Routed SOAP Messages

- Why WS-Addressing & WS-Security?



- Multiple "hops" for a message to get from A to B. A segment might not be HTTP. A "router" may chose to persist the message. An ESB is typically based on mediators so that a consumer doesn't not know the endpoint.

WS-* (JBoss AS 4.0.4 via JBossWS 1.0)

- WS-Policy
  - ✓ A standard XML schema for identifying:
    - Digital signature required
    - Encryption required
    - Security token to be used (x509, user/pass)
    - Expiration of the message
    - User and application priorities
    - Traffic control/flow
  - ✓ Allows for automatic enforcement of the above
- MTOM
  - ✓ Message Transmission Optimization Mechanism
  - ✓ Provides an encoding mechanism for including binary attachments directly in the SOAP envelope. Similar in concept to MIME and email attachments.

WS-* (JBoss Transactions)

- WS-Context
  - ✓ Provides SOAP header extensions that provides session management (similar to a HTTP cookie)
- WS-Coordination
  - ✓ A generic coordination infrastructure
- WS-AtomicTransaction
  - ✓ 2PC – two phase commit coordination (builds on WS-Coordination)
- WS-BusinessActivity
  - ✓ Long running business activities, using forward compensation-based transactions (builds on WS-Coordination)

SOAP Headers

- Why?
- What?
- How?

BPEL Introduction

BPEL Introduction

- Business Process Execution Language
- W3C Status – expected to be ratified in November of 2006
- JBoss Status – based on jBPM, currently Alpha4
- BPEL "fixes" WSDL
  - ✓ WSDL operations are unordered. The consumer has no indication what comes first.
  - ✓ Supports ordering/sequencing
  - ✓ Supports concurrency
  - ✓ Supports choreography with external entities

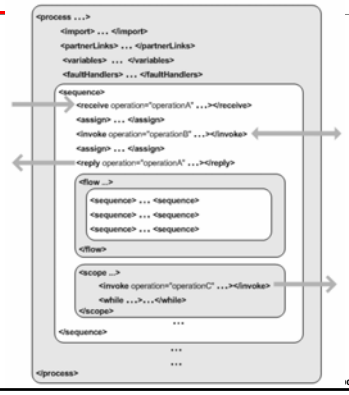
BPEL Introduction II

- XML-based syntax
- A BPEL process has a WSDL and is a web service producer itself.
- It consumes other WSDLs
- Defines the process flow/conversation/ interaction with the end consumer.
- Allows for "stateful" behavior
- Allows for asynchronous invocation of other web services (forking) and eventual joining.
- A process instance is a particular conversation with a particular consumer.
- A BPEL engine should allow for multiple concurrent conversations.

BPEL Keywords

- <invoke> - call another webservice
- <receive> - an exposed "endpoint" for external invocation
- <reply> - generating a response
- <wait> - wait N amount of time
- <assign> - copy a variable from one place to another
- <throw> - throw an exception
- <catch> - catch the possible faults that are identified in a WSDL
- <terminate>/<exit> - get out
- <empty> - do nothing
- <sequence> - ordered series of steps
- <switch> - case statement enables conditional branching, deterministic
- <while> - loop
- <pick> - is based on message - message based routing
- <flow> - parallel processing
- <onMessage> - uses correlationSets that allow a callback from a partner webservice, we handle that via a JMS queue under the covers and use a message selector

.bpel file



Example of BPEL

```
<process>
 <partnerLinks>
    <!-- realizes the abstract relationship with the caller -->
    <partnerLink name="caller" partnerLinkType="tns:helloPLT" myRole="service"/>
  </partnerLinks>

  <variables>
    <!-- holds the incoming message -->
    <variable name="request" messageType="tns:nameMessage"/>
     <!-- holds the outgoing message -->
    <variable name="response" messageType="tns:greetingMessage"/>
  </variables>

  <sequence>
    <!-- receive a message carrying the name of a person -->
    <receive operation="sayHello" partnerLink="caller" portType="tns:helloPT"
     variable="request" createInstance="yes"/>
    <!-- compose a greeting phrase containing the name -->
    <assign>
      <copy>
        <from expression="concat('Bonjour, ',
                          bpel:getVariableData('request', 'name'), '!')"/>
        <to variable="response" part="greeting"/>
      </copy>
    </assign>
    <!-- reply with a message carrying the greeting -->
    <reply operation="sayHello" partnerLink="caller" portType="tns:helloPT"
     variable="response"/>
  </sequence>
</process>
```


BPEL Demonstration

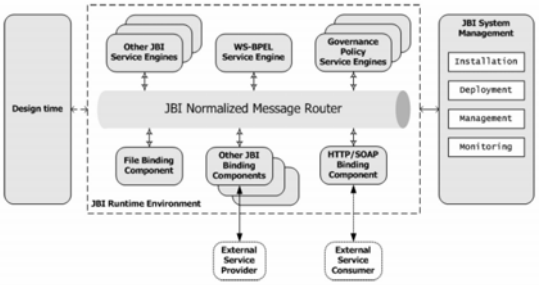
- Basic Hello World
- With VB.NET Consumer

- Considerations:
  - ✓ Alpha4
  - ✓ Presently runs on 4.0.3 - NOT 4.0.4.GA
  - ✓ Examples use the older WS4EE process for building the Web Services it needs to consume and produce (instead of the new 181 annotated web services available on JBoss App Server 4.0.4.GA)

JBI

Java Business Integration

JBI – Java Business Integration (JSR 208)



Java EE 5 would be a Service Engine. JBI is important because you may get multiple vendors to deliver BCs and SEs so you can plug & play with different implementations. A business app developer only needs to build 181-annotated POJOs.

Source: Implementing SOA w/JavaEE 5 - Sun Microsystems