This presentation is for informational purposes only and may not be incorporated into a contract or agreement.
This document is for informational purposes. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in this document remains at the sole discretion of Oracle. This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. This document and information contained herein may not be disclosed, copied, reproduced or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.
THE INFORMATION COMPANY
Real Interoperability Between J2EE™ and .NET Web Services: Practical Advice for Developers
Agenda With Section Highlights

Problem Domain
Fixing Interoperability Problems
  Changing the Server
  Changing the WSDL
  Changing the Client
Best Practices – Top Down Design
Top 3 Take Aways
Problem Domain
Fixing Interoperability Problems
  Changing the Server
  Changing the WSDL
  Changing the Client
Best Practices – Top Down Design
Top 3 Take Aways
Two SOAP Stacks? N SOAP Stacks.

Java

- Apache SOAP
- Apache Axis 1.1/1.2
- Oracle JAX-RPC
- Sun JAX-RPC
- BEA JAX-RPC
- IBM JAX-RPC

.NET

- Microsoft SOAP Toolkit 3.0
- .NET Framework 1.1
- .NET Framework 2.0
- WSE 1.1
- WSE 2.0
- WSE 3.0
- Indigo
## Two Scenarios – Replayed Over and Over Again

<table>
<thead>
<tr>
<th>Developer builds Java class</th>
<th>Developer builds C# class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Publishes it bottom up as Web service</td>
<td>Publishes it bottom up as Web service</td>
</tr>
<tr>
<td>Points .NET at WSDL</td>
<td>Points Java tooling at WSDL</td>
</tr>
<tr>
<td>Builds C# client</td>
<td>Builds Java client</td>
</tr>
<tr>
<td>Runs client</td>
<td>Runs client</td>
</tr>
<tr>
<td>ERROR</td>
<td>ERROR</td>
</tr>
</tbody>
</table>
Industry Backed Mechanisms to Solve the Problem with Varying Success

- **WS-I**
  - Profiles (BP, BSP, AP)
  - Sample application
  - Testing

- **SOAPBuilders**
  - WSDL 1.1
  - SOAP 1.1
  - RPC/encoded, Doc/literal
The Problem

- Achieving interoperability with legacy and modern SOAP stacks
- Both pre-WS-I and post-WS-I stacks have interoperability issues
- WS-I compliance in itself does not guarantee interoperability
- Serialization and WSDL generation idiosyncrasies with various Microsoft and Java stacks
First Need to Understand the Process Happening in Each Runtime

Client

C#

Application Code

Serialization

Deserialization

Request

Response

.NET Framework

XML

HTTP

Service

Java

Application Code

JAX-RPC

WSDL

Deserialization

Serialization

Request

Response

XML

HTTP

ORACLE
### Message Formats

- **Next Need to Understand Message Formats Used in Both Worlds**

<table>
<thead>
<tr>
<th>Document literal (a la WS-I mode, a.k.a wrapped style)</th>
<th>Document Literal (e.g. bare style, like message style with Axis)</th>
<th>RPC Literal (mirror analog of document literal wrapped)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPC Encoded (limited to SOAP encoding rules – Section 5)</td>
<td>RPC Encoded Untyped (e.g. MSFT)</td>
<td>RPC Encoded, with proprietary extensions (e.g. collections)</td>
</tr>
</tbody>
</table>
Message Format Survey

- From XMethods (http://www.xmethods.com) in 2005

Source: http://www.xmethods.net/inspection.wsil - Friday, May 13, 2005
Lastly Need an Approach to Solving Interoperability Issues

- Solve the problem before you start
  - Start with interoperability in mind
  - Follow known patterns that work
    - Top down or well known bottom up approaches
- Solve the problem after the fact
  - Change the service implementation
  - Change the deployment options of the service
  - Change the WSDL published [server side]
  - Edit a local copy of the WSDL [client workaround]
  - Edit the generated code [client hack]
Agenda With Section Highlights

Problem Domain

**Fixing Interoperability Problems**
- Changing the Server
- Changing the WSDL
- Changing the Client

Best Practices – Top Down Design

Top 3 Take Aways
Why Change the Server?

- Message formats don’t match
  - Doc/lit talking to rpc/encoded
  - Rpc/encoded no types talking to rpc/encoded
  - Solution: Change the server configuration to match

- Unsupported Datatypes
  - .Net datasets, Java webrowsets
  - Java collections
  - Nillable values in arrays
  - Solution: Change the server implementation to use supported types across both implementations

- Problematic WSDL generation
  - SOAP action
  - Solution: Change server implementation to fix generated WSDL
A Simple Scenario: Versioning an Add Service Over Time

Calling 3 endpoints from C# and Java, what to expect?

1) Generates both clients from the first Service

- C# Math Client
- Java Math Client

2) Both clients call each service in turn.

- MathService1.asmx
- MathService2.asmx
- MathService3.asmx
DEMONSTRATION

Use case #1
Lessons Learned from Changing the Server

- A given client work best only with one service
  - Plan for versioning of both server and client
  - Version your WSDL, so that you can compare them
- Interchangeable services must implement the same contract, as defined by the WSDL
- Changes to publish services can easily break existing clients
- Different platforms have different way to handle a deserialization problem [MathService2.asmx].
- Use an empty value (""”) for SOAPAction
Problem Domain

**Fixing Interoperability Problems**
- Changing the Server
- **Changing the WSDL**
- Changing the Client

Best Practices – Top Down Design

Top 3 Take Aways
Why Change the WSDL?

- WSDL is generally independent of the message exchange and serialization
- It is the contract both sides implicitly agree to adhere to
- As a result, it is used typically to generate an appropriate client
- If the message on the wire does not match the WSDL, an incorrect client will be generated
- Solution: Fix the WSDL
Common WSDL Edits

- xsd:any versus xsd:anyType
- import missing locations
- Unspecified namespace prefix
- Re-using different prefix for the same namespace
- xsd:boolean vs. soapenc:boolean
- SOAP Arrays
  - maxOccurs="unbounded" not "*"
  - minOccurs="0" unless array can not be empty
Diagnosing WSDL Errors

- Use tools like SOAPScope
Example 1: XML Schema – xsd:any Wildcard

Common Issues in Authoring/Generating XML Schema

• Using any as a type, inside a schema
  
  `<element name="QuotesResponse" type="any"/>

• Using any as a type for a part of a message

  `<message name="QuotesResponse">
   
   `<part name="result" type="xsd:any"/>

  </message>`

• Using any as a named element, for an element attribute

  `<message name="QuotesResponse">
   
   `<part name="result" element="xsd:any"/>

  </message>`
Example 1: XML Schema – xsd:any Wildcard

Two Ways to Resolve This by Changing the WSDL

- Uses the correct XML type: anyType
  
  ```xml
  <element name="QuotesResponse" type="anyType"/>
  <message name="QuotesResponse">
    <part name="result" type="xsd:anyType"/>
  </message>
  ```

- Uses any as a wildcard, in place of a named element in a complex type
  
  ```xml
  <element name="result" nillable=true" minOccurs="0"
...
  <complexType>
    <sequence>
      <any/>
    </sequence>
  </complexType>
  ```
Example 2: XML Schema – Import and Location

Often Used as a Way to Reference Another Schema

- Here is an example where the location attribute is missing from the xsd:import.

  ```xml
  <schema xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:apachesoap="http://xml.apache.org/xml-soap"
    targetNamespace="http://message.samples"
    elementFormDefault="qualified">
    <import namespace="http://xml.apache.org/xml-soap"/>
    <element name="elem" type="apachesoap:Element" />
  </schema>
  ``

- Assuming that a WSDL processor will recognize the namespace is wrong
Example 2: XML Schema – Import and Location

Two Ways to Resolve This by Changing the WSDL

• If you know where to find the schema, add the location

  <schema xmlns="http://www.w3.org/2001/XMLSchema"
      xmlns:apachesoap="http://xml.apache.org/xml-soap"
      targetNamespace="http://message.samples"
      elementFormDefault="qualified">
    <import namespace="http://xml.apache.org/xml-soap"
        location="http://xml.apache.org/some-url.xsd" />
    <element name="elem" type="apachesoap:Element" />
  </schema>

• Otherwise, remove the import and use a replacement type

  <element name="elem" type="anyType" />

Demonstration

Building a .Net client to Java WS
Lessons Learned from WSDL Editing

• Understand the scoping rule of namespace
  • Be careful about copying of namespace
• Use different tool kits to validate the WSDL
  • One tool is not a guarantee of interoperability
  • XMLSpy is not good enough!
• Be careful about the version of schema
  • Always use 2001 but still see 1999
  • WSDL specification uses 10/2000! Don’t use it!
• Be careful in the usage of XML schema types
  • Choice, group, redefine …
Agenda With Section Highlights

Problem Domain

**Fixing Interoperability Problems**
- Changing the Server
- Changing the WSDL
- **Changing the Client**

Best Practices – Top Down Design

Top 3 Take Aways
Why Change the Client?

- Client generation fails
- Client generation works but fails at runtime
- No access to the server implementation
- No access to the server WSDL

Solution:
- Change the WSDL locally and re-generate client
- Change the client
Example: I am Getting a Null Back … Why?

- Generation of the C# Client
- Inspect the client code
- Output of the program
- Capture the request/response
- Look at the payload for the response
- Compare the payload with the WSDL
- Change the generated code to work around
DEMONSTRATION

The "Real" Interoperability
Lessons Learned from Changing the Client

Why do I get a null value back?

• Check the format of the response on the wire
  • The server is sending an empty response
    • The most likely issue is a failure to deserialize the request
  • The server replies with the correct response
    • The most likely issue is a failure to deserialize the response
• The bottom line is that the message on the wire do not adhere to the Schema advertised in the WSDL.
Agenda With Section Highlights

Problem Domain
Fixing Interoperability Problems
  Changing the Server
  Changing the WSDL
  Changing the Client

Best Practices – Top Down Design

Top 3 Take Aways
Best Practices: Top Down Design - WSDL to Java
Typical Approach

- Author XML schema for inbound and outbound messages
  - Frequently companies will maintain a schema dictionary
- Use prototypical WSDL and set message formats to authored XML schema
  - Use a WS-I compliant template
- Choose your data binding
  - SAAJ, JAXB or JAX-RPC bindings
Primary Issues with Top Down

- Complexity
  - Tooling for top down is limited
    - .NET extensions from 3rd parties
    - Java – Oracle, Sun RI, Axis(?)
- Indigo
  - Contract driven from code is way forward
- JAX-WS
  - Unclear preferred mechanism going forward
DEMONSTRATION

Top Down Web Service
Agenda

Problem Domain
Fixing Interoperability Problems
  Changing the Server
  Changing the WSDL
  Changing the Client
Best Practices – Top Down Design

Top 3 Take Aways
Top 3 Take Aways

• Make sure you test your service with multiple Web Service toolkits
• Make sure you have a tool for capturing messages on the wire
  • SOAPScope, JDeveloper, WSE, IBM, BEA, Axis ...
• Top down design will maximize your interoperability chances
  • Ask your vendor for their tooling!
Agenda With Section Highlights

✓ Problem Domain
✓ Fixing Interoperability Problems
✓ Changing the Server
✓ Changing the WSDL
✓ Changing the Client
✓ Best Practices – Top Down Design
✓ Top 3 Take Aways
QUESTIONS & ANSWERS
ORACLE FUSION MIDDLEWARE

Learn the Technology
Visit OTN Microsites: otn.oracle.com

Try the Software
Visit the Hands-On Labs in Moscone West 1003, 1004

Ask Our Experts
Fusion Middleware Demogrounds; Sessions