



Developing Interoperable Web Services for the Enterprise

Simon C. Nash

IBM Distinguished Engineer
Hursley, UK

nash@hursley.ibm.com



What is a Web Service?

Is it:

- a service offered *via* the Web?
- XML data interchange?
- SOAP messaging over HTTP?
- something described using WSDL?
- all of the above?

W3C Proposed Definition of Web Service



A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

W3C Proposed Definition of Web Service

A Web service is a software system designed to support **interoperable machine-to-machine interaction** over a network. It has an interface described in a machine-processable format (specifically **WSDL**). Other systems interact with the Web service in a manner prescribed by its description using **SOAP-messages**, typically conveyed using **HTTP** with an **XML serialization** in conjunction with other Web-related standards.



Disclaimer

- This is not an in-depth tutorial on the WSDL, SOAP, UDDI, WS-Security, JAX-RPC or JSR 109 specs
- It is a hands-on guide to how to develop interoperable Web services in Java
- Denise Hatzidakis has two talks covering the Web services specifications



Agenda

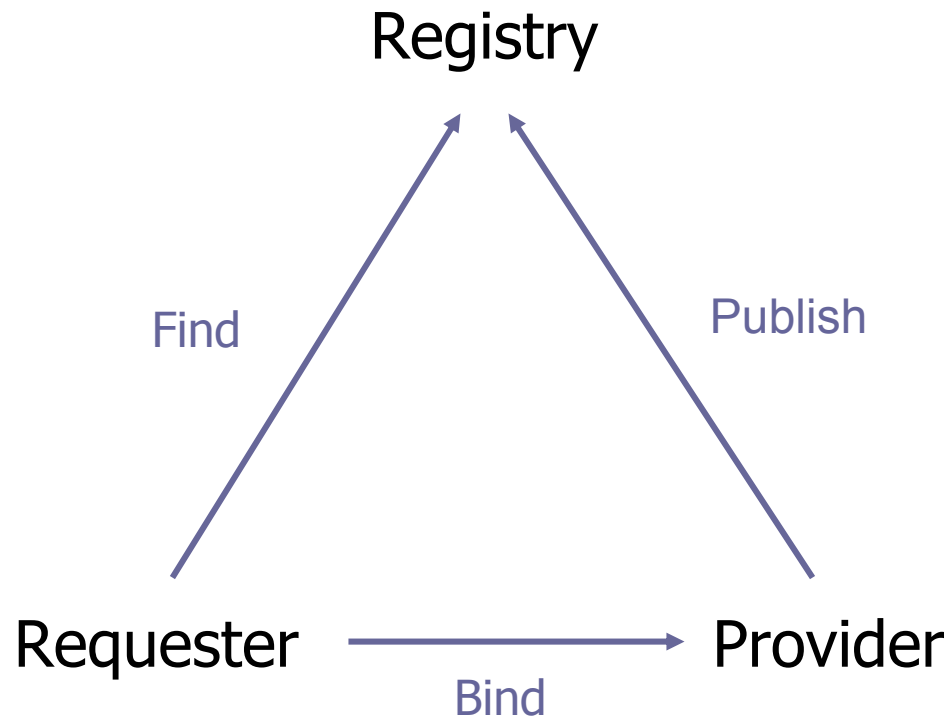
- WSDL, SOAP, UDDI
- JAX-RPC
- Web Services for J2EE (JSR 109)
- WS-Security
- Web services interoperability
- Web Services Interoperability Organization (WS-I.org)



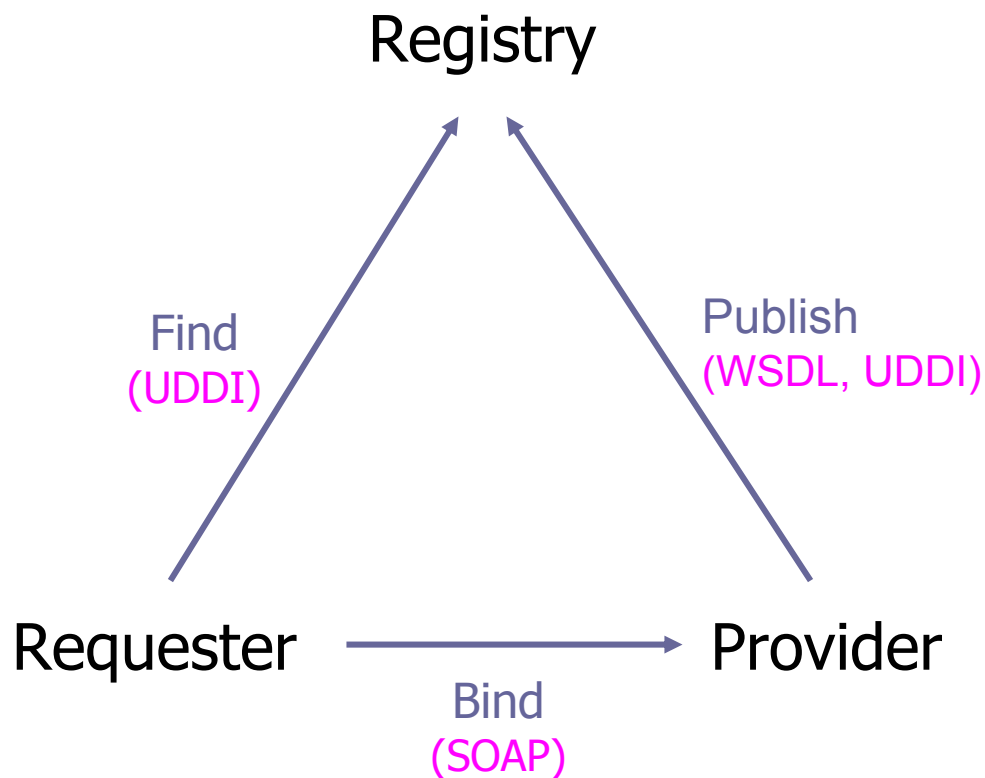
Acronym Soup

- WSDL = Web Services Description Language
 - “IDL for Web services”
- SOAP = Simple Object Access Protocol
 - XML-based service invocation protocol
- UDDI = Universal Description, Discovery and Integration
 - protocol for publishing and finding Web services

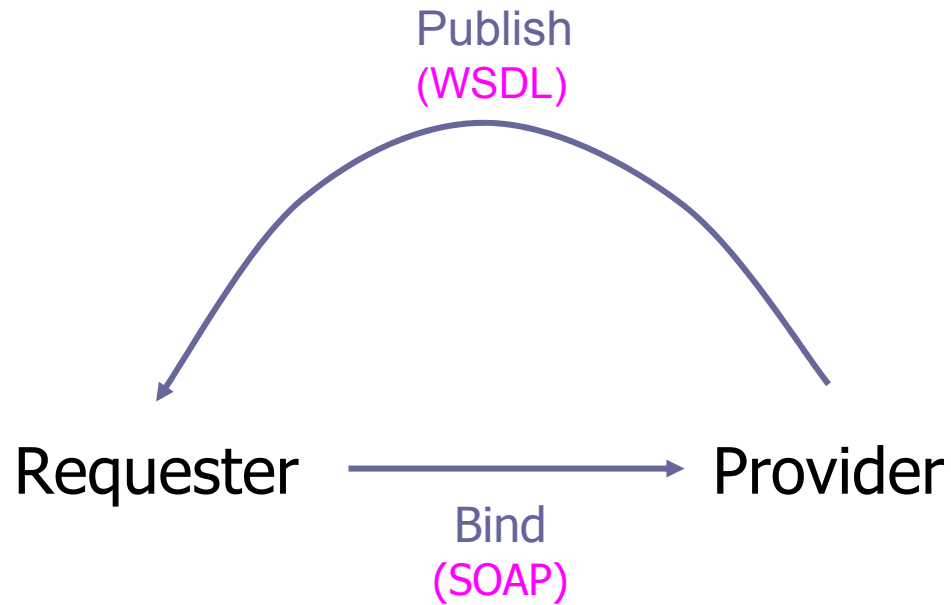
Web Services Roles



Web Services Specifications



Without a Registry



WSDL Example: The StockQuote Web Service

- Single operation: getPrice
- Input parameter: String
- Output result: float
- Historical prices (August 2003)
- Extremely limited stock selection



StockQuote.wsdl (1 of 3)

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://demo.ibm.com" xmlns=
"http://schemas.xmlsoap.org/wsdl/" xmlns:intf="http://demo.ibm.com" xmlns:wsdl=
"http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap=
"http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<wsdl:types>
  <schema targetNamespace="http://demo.ibm.com"
    xmlns="http://www.w3.org/2001/XMLSchema">
    <element name="getPrice">
      <complexType>
        <sequence> <element name="symbol" type="xsd:string"/> </sequence>
      </complexType>
    </element>
    <element name="getPriceResponse">
      <complexType>
        <sequence> <element name="getPriceReturn" type="xsd:float"/> </sequence>
      </complexType>
    </element>
  </schema>
</wsdl:types>
... see next slide
</wsdl:definitions>
```



StockQuote.wsdl (2 of 3)

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
... see previous slide
```

```
<wsdl:message name="getPriceRequest">
```

```
  <wsdl:part element="intf:getPrice" name="parameters"/>
```

```
</wsdl:message>
```

```
<wsdl:message name="getPriceResponse">
```

```
  <wsdl:part element="intf:getPriceResponse" name="parameters"/>
```

```
</wsdl:message>
```

```
<wsdl:portType name="StockQuote">
```

```
  <wsdl:operation name="getPrice">
```

```
    <wsdl:input message="intf:getPriceRequest" name="getPriceRequest"/>
```

```
    <wsdl:output message="intf:getPriceResponse" name="getPriceResponse"/>
```

```
  </wsdl:operation>
```

```
</wsdl:portType>
```

```
... see next slide
```

```
</wsdl:definitions>
```



StockQuote.wsdl (3 of 3)

```
<?xml version="1.0" encoding="UTF-8"?>
  ... see previous slide
  <wsdl:binding name="Demo1SoapBinding" type="intf:StockQuote">
    <wsdlsoap:binding style="document"
      transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="getPrice">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getPriceRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getPriceResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>

  <wsdl:service name="StockQuoteService">
    <wsdl:port binding="intf:Demo1SoapBinding" name="Demo1">
      <wsdlsoap:address location="http://localhost:6080/Demo/services/Demo1"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```



StockQuote Implementation

```
package com.ibm.demo;
import java.util.Hashtable;

public class Demo1 {
    private Hashtable stocks = new Hashtable();
    public Demo1() {
        stocks.put("IBM", "81.49");
        stocks.put("MSFT", "25.61");
        stocks.put("SUNW", "3.62");
        // add your favourite companies here
    }

    public float getPrice(String symbol) {
        return Float.parseFloat((String)stocks.get(symbol));
    }
}
```



StockQuote Sample Client

```
package com.ibm.demo;
import java.net.URL;
import javax.xml.namespace.QName;
import javax.xml.rpc.Service;
import javax.xml.rpc.ServiceFactory;

public class Demo1Client {
    public static void main(String[] args) {
        String wsdlURL = "http://localhost:6080/Demo/services/Demo1?wsdl";
        String namespace = "http://demo.ibm.com";
        String serviceName = "StockQuoteService";
        String portName = "Demo1";
        try {
            ServiceFactory factory = ServiceFactory.newInstance();
            Service myService = factory.createService(
                new URL(wsdlURL), new QName(namespace, serviceName));
            StockQuote endpoint = (StockQuote)myService.getPort(
                new QName(namespace, portName), StockQuote.class);
            System.out.println("IBM price "+endpoint.getPrice("IBM"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

StockQuote Service Endpoint Interface

```
// The following interface can be generated automatically from the
// Demo1 implementation

package com.ibm.demo;

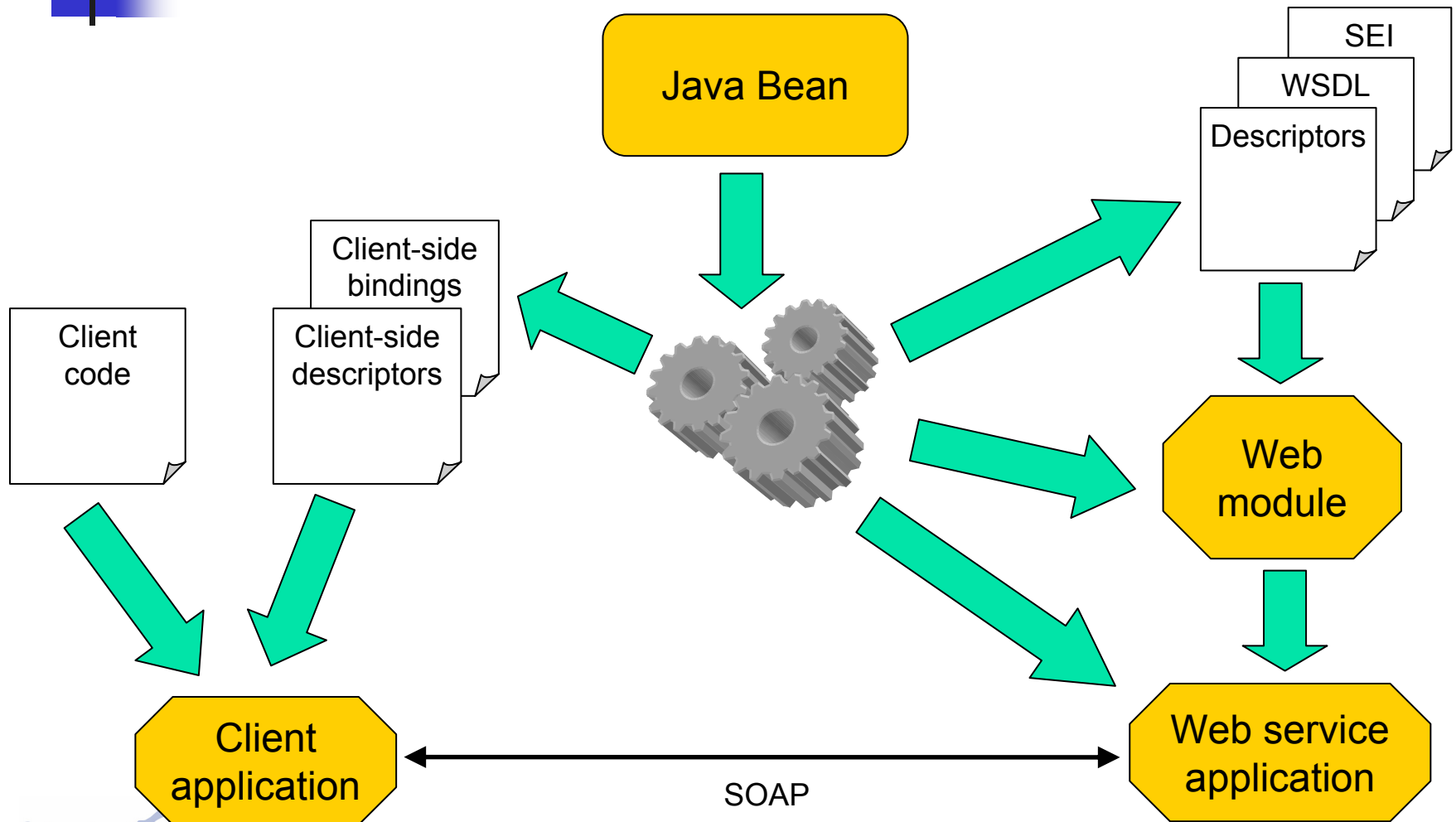
public interface StockQuote extends java.rmi.Remote {
    public float getPrice(String symbol) throws java.rmi.RemoteException;
}
```



StockQuote Demo

- Generate WSDL from JavaBean
- Generate servlet implementing the StockQuote Web service
- Generate client-side bindings (optional)
- Deploy implementation to application server
- Build client
- Run client
- Show the SOAP messages being exchanged

StockQuote Demo





A Word on Developer Tools

- There are lots of them!
- from IBM, BEA, Sun, Oracle, Borland, The Mind Electric, Systinet, Apache, Eclipse, *etc.*
- I had to choose one for the demos
- I chose the IBM WebSphere SDK for Web Services (WSDK)



JAX-RPC

- Java API for XML-based RPC
- JCP specification (JSR 101)
- Java <-> WSDL/XML mappings
- Java client APIs for invoking Web services
- Servlet-based endpoints for Web services
- SOAP bindings
- Other: attachments, message handlers, type mappings, interoperability

Web Services for J2EE (JSR 109)

- Based on (and includes) JAX-RPC
- Includes support for
 - Web services deployed in J2EE containers
 - J2EE components as Web services clients
 - JNDI lookup for Web services
 - Interoperation with non-J2EE implementations
- Can be implemented on J2EE 1.3
- An integral part of J2EE 1.4



JSR 109 Specification Overview

- Server support:
 - J2EE Web container, EJB container (stateless session bean only)
- Client support:
 - J2EE Web container, EJB container, J2EE application client container
- Programming model rules
- Standard deployment descriptors



JSR 109 Development Models

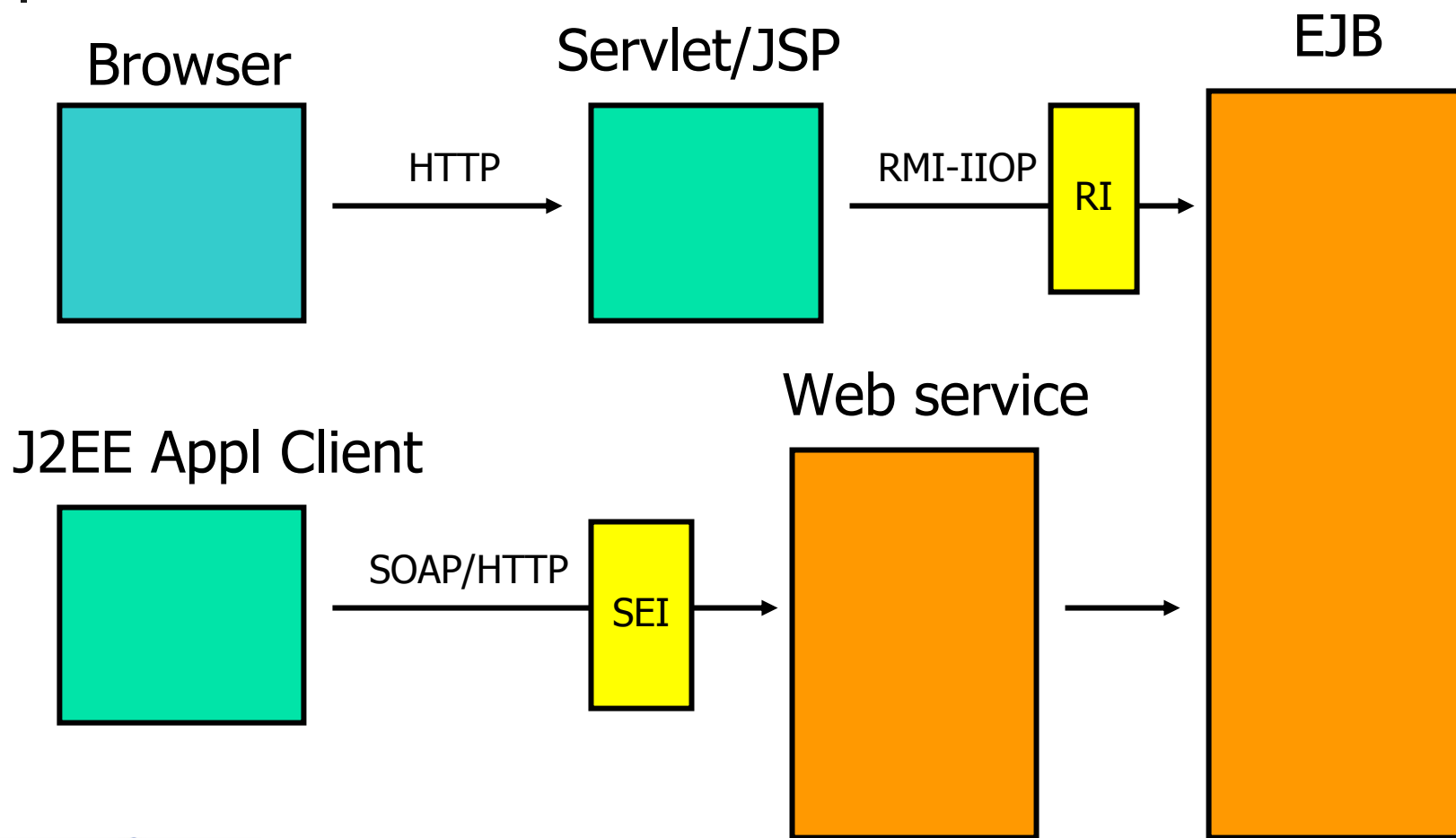
- Top down: Create Java implementation to conform to existing WSDL definition
- Bottom up: Generate WSDL definition from existing Java class or stateless session EJB
- Other:
 - create Java Service Endpoint Interface and implementation, generate WSDL from the SEI
 - generate WSDL from Java code, edit the WSDL



AddressBook Demo

- Show original session EJB, servlet/JSP client
- Generate WSDL from session EJB
- Generate servlet and EJB implementing the AddressBook Web service
- Generate client-side JSR 109 bindings
- Deploy Web service to application server
- Build Web services client
- Run Web services client

AddressBook Demo





WS-Security

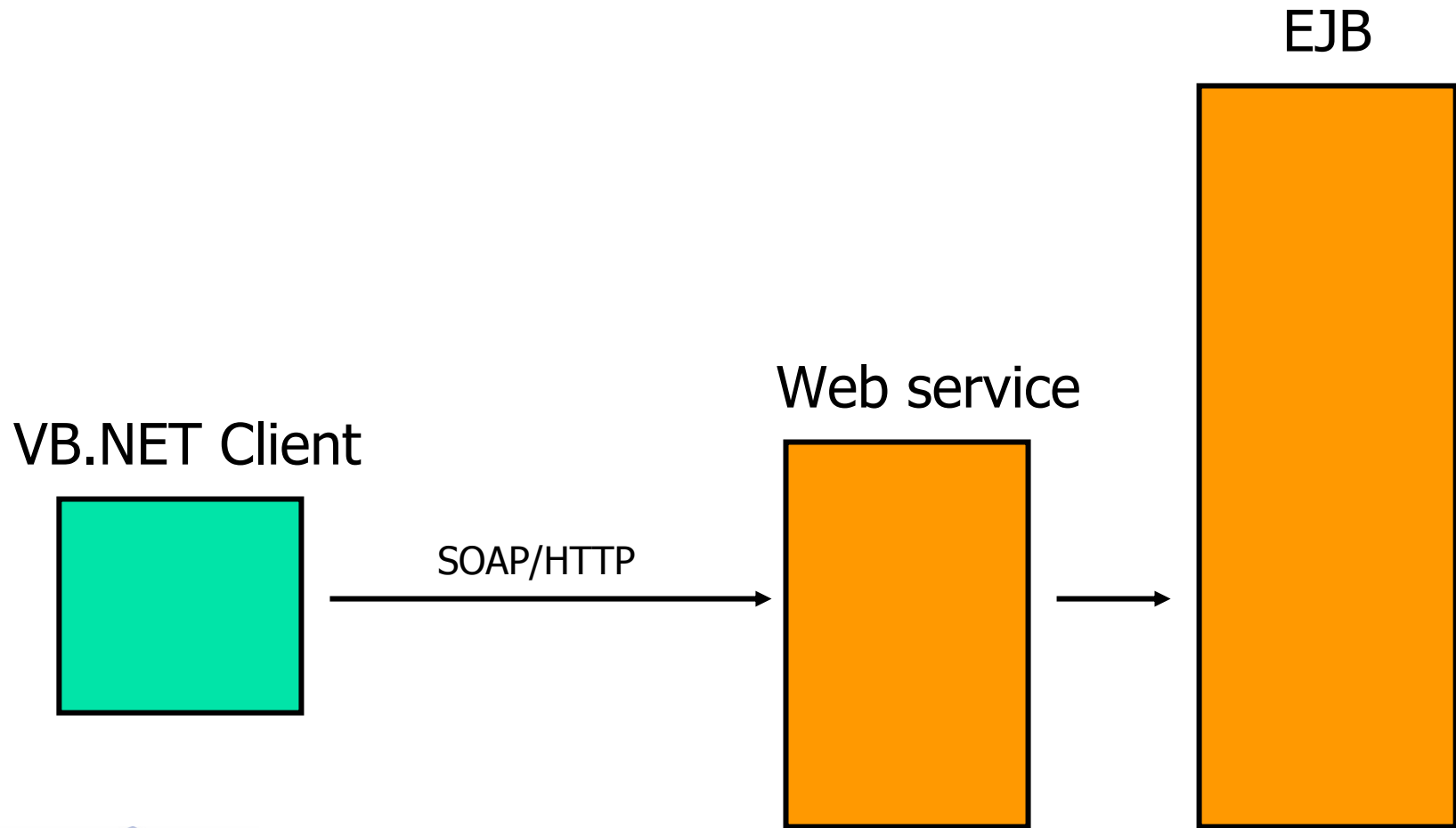
- JSR 109 has HTTP basic auth and HTTPS
- WS-Security adds message-level security for transport-independent end-to-end security
 - Encryption
 - Digital signature
 - Security token
 - Timestamp
- Configuration still implementation-specific



Web Services Interoperability

- Interoperability is fundamental to Web services
- Interoperability doesn't just happen
- Many bilateral and multilateral efforts (*e.g.*, SOAPBuilders) to test and improve interoperability
- WS-I.org was formed to promote Web services interoperability

AddressBook Interoperability Demo





What Is WS-I.org?

- Industry organization with Web services vendors and users as members
- Not a standards body
- Produces materials to support interoperability
 - Profiles
 - Test tools
 - Usage scenarios
 - Sample applications



WS-I Basic Profile 1.0

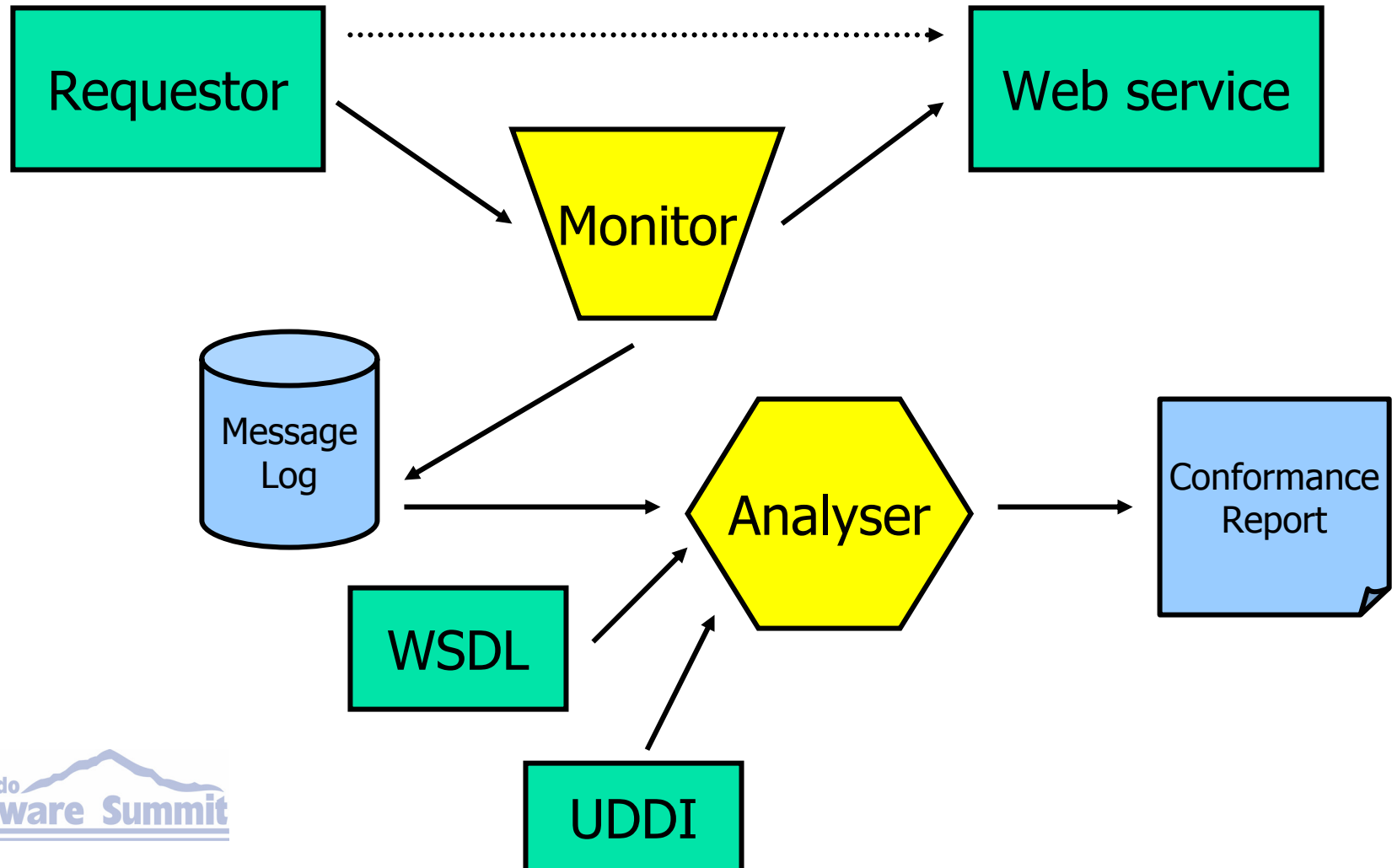
- Final version released on August 12, 2003
- Based on SOAP 1.1, HTTP 1.1, WSDL 1.1, XML 1.0, XML Schema 1.0, UDDI V2
- Contains clarifications/amendments to the above specs that affect interoperability
- Attachments support deferred to Basic Profile 1.1



WS-I test Tools

- Monitor communications between a Web service and a requestor
- Analyse SOAP messages, WSDL documents, and UDDI entries for conformance to the Basic Profile
- Produce a conformance report (summary and detail)

WS-I test Tools Flow





WS-I test Tools Demo

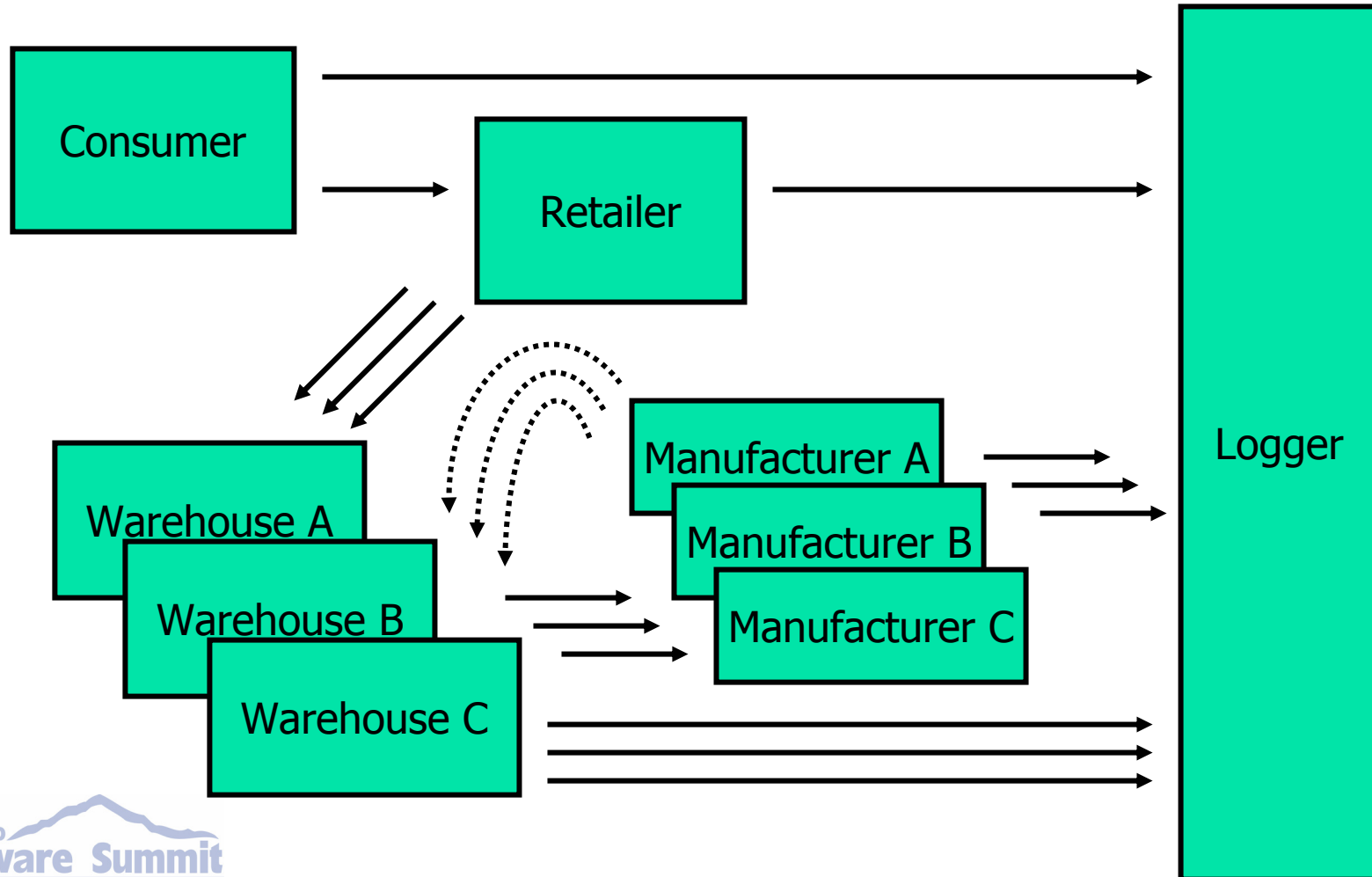
- Reconfigure the AddressBook Web service to pass messages through the monitor
- Create the message log
- Run the analyzer to create the conformance report
- Introduce a conformance violation to show how errors are reported



WS-I Sample Application

- Supply Chain Management application
- Designed to show all the WS-I Basic profile 1.0 usage scenarios
 - Request/response
 - One way
 - Callback

Sample Application Demo





WS-I Future Directions

- Basic Security Profile
 - Transport security
 - SOAP messaging security
 - Additional considerations for Basic Profile
 - Develop usage scenarios
 - Liaise with OASIS WSS TC
 - Based on HTTPS, S/MIME, Cryptographic Message Syntax, OASIS Web Services Security



References

- WSDL: <http://www.w3.org/TR/wsdl.html>
- SOAP: <http://www.w3.org/TR/SOAP/>
- UDDI: <http://www.uddi.org>
- JAX-RPC: <http://www.jcp.org/en/jsr/detail?id=101>
- JSR 109: <http://www.jcp.org/en/jsr/detail?id=109> and <http://www.jcp.org/en/jsr/detail?id=921>
- WS-Security: <http://www.ibm.com/developerworks/library/ws-secure/>
- WSDK: <http://www.ibm.com/developerworks/webservices/wsdk>
- WS-I: <http://www.ws-i.org>



Summary

- Web services are moving from emerging to adoption phase
 - Base standards are in place, many others still being worked
 - Interoperability is today's focus, but will soon be routine



Questions?

