Developing real world Web services using J2ME™, J2SE™, J2EE™

Sang Shin, Carol McDonald
Java™ Technology Evangelists

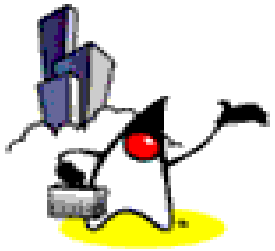
sang.shin@sun.com, carol.mcdonald@sun.com

Sun™ Tech Days

Agenda

- Web Services architecture over J2EE
- JAX-RPC architecture
- How to develop JAX-RPC based Web Service
- JAX-RPC client programming model
- SOAP message handler
- Document-driven model
- WS-I and Web Services interoperability
- Changing landscape of Web Services
  - ebXML, Fast Web service, Metadata Web service, Orchestration, Transaction, Reliable messaging, Security

Web Services Architecture over J2EE



Sun microsystems

Push your development further

What Is a J2EE Web Service?

- A set of endpoints (ports) operating on messages
- Ports are operating within a container
  - Container provides runtime environment
  - Contract for runtime environment are specified in JAX-RPC, EJB 2.1, JSR 109 (Web Services for J2EE)
- Service is described in WSDL document and published to a registry
  - WSDL specifies a contract between service provider and client

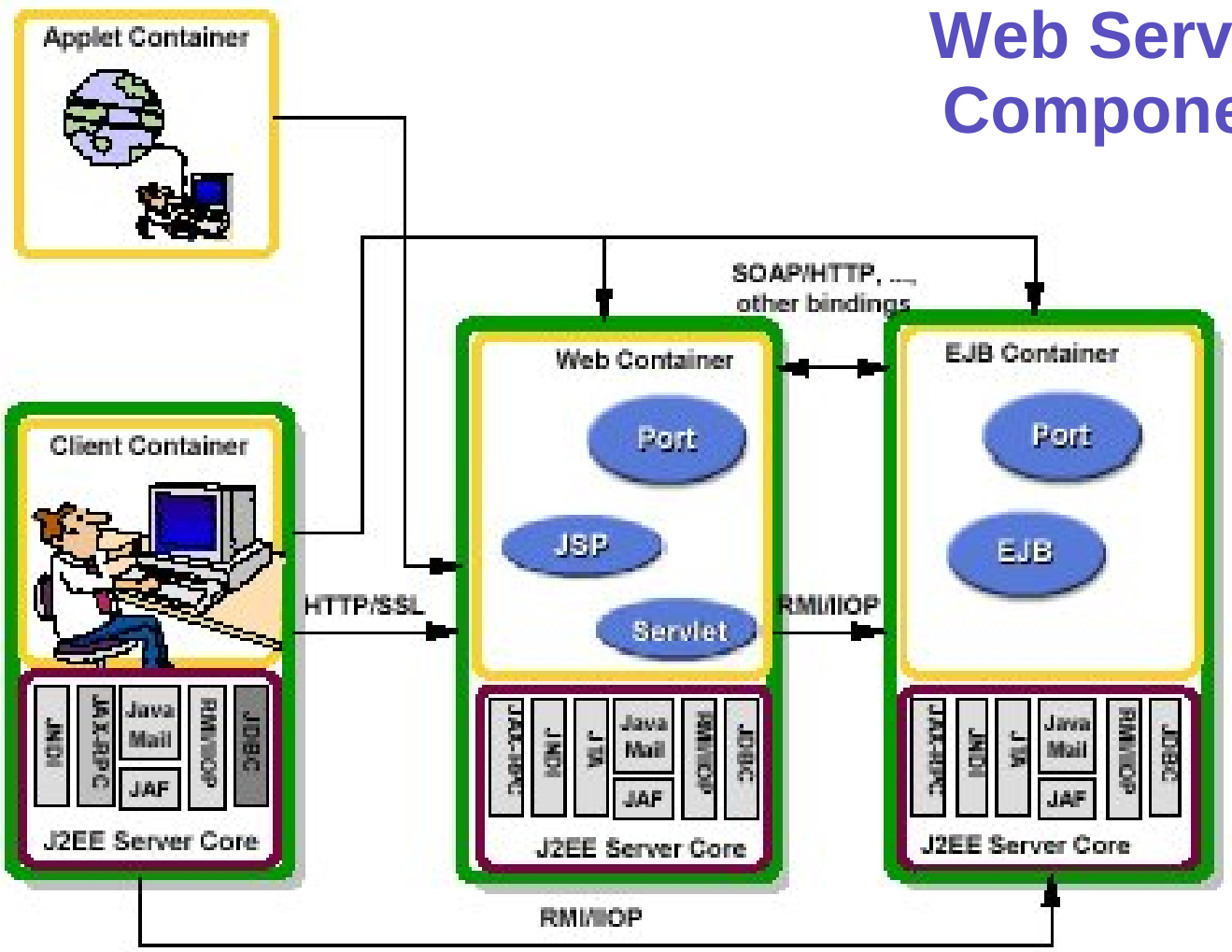
Web Service Component and Container

- Container vs. Component model
  - Web Services components get executed within a container
  - Web Services components are portable (under J2EE 1.4)
- Web Service components
  - Web-tier (Servlet-based endpoint)
  - EJB-tier (Stateless session bean-based endpoint)

Service Endpoint Implementation

- Web service endpoint realized as either:
  - Servlet-based endpoint
  - Stateless session bean
- JAX-RPC 1.1 specifies Servlet-based endpoint model
- JSR-109 and EJB™ 2.1 technology specify stateless session bean endpoint model

Web Service Components



Source: Web Services for J2EE (JSR 109), V1.0

JAX-RPC Architecture

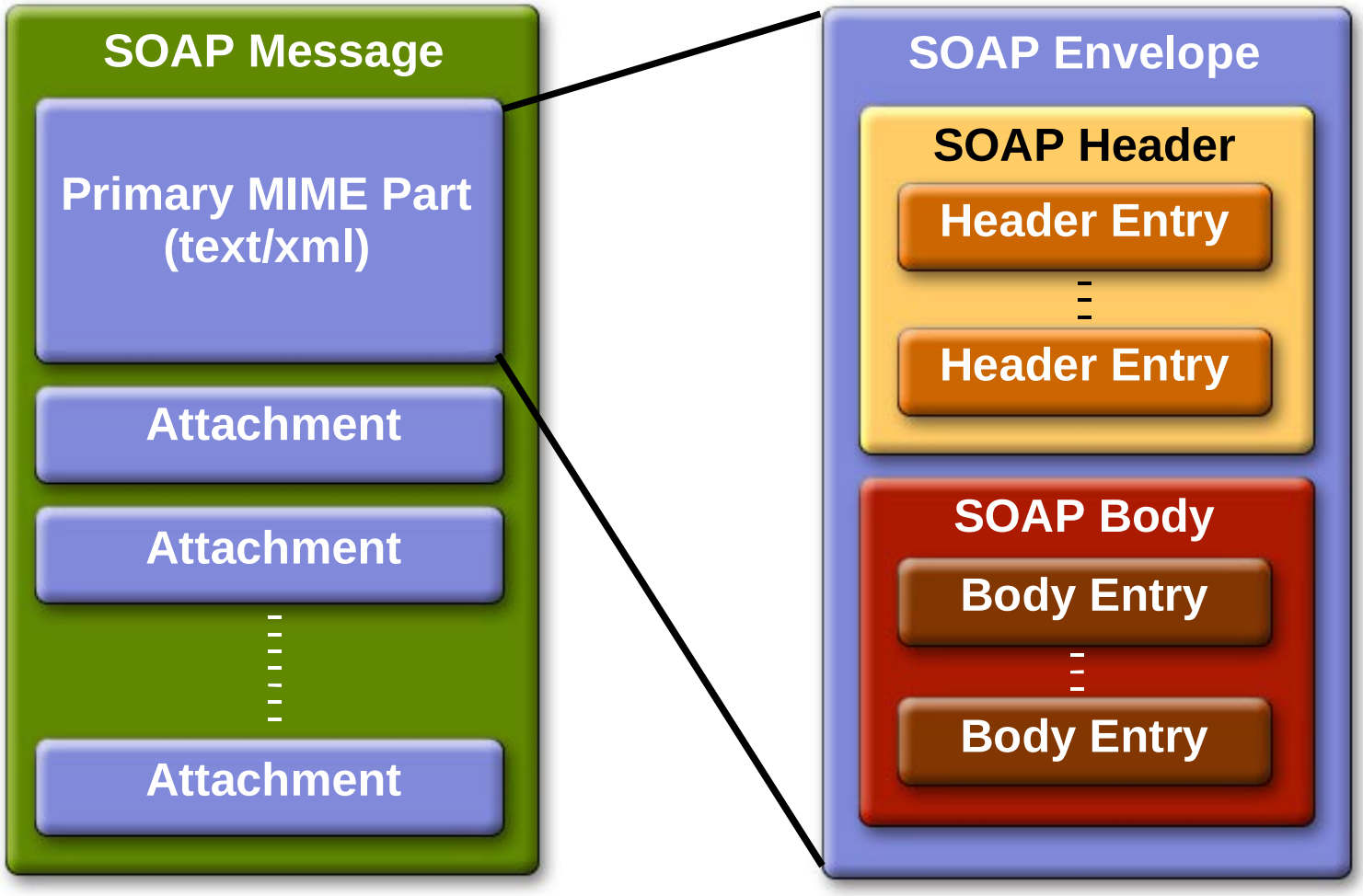
JAX-RPC Features

- Servlet-based Web service endpoint model
- WSDL to/from Java™ mapping
- XML data types to/from Java™ types mapping (serialization)
- Extensible type mapping
- SOAP Message Handler framework
- Packaging
- Client Programming Models

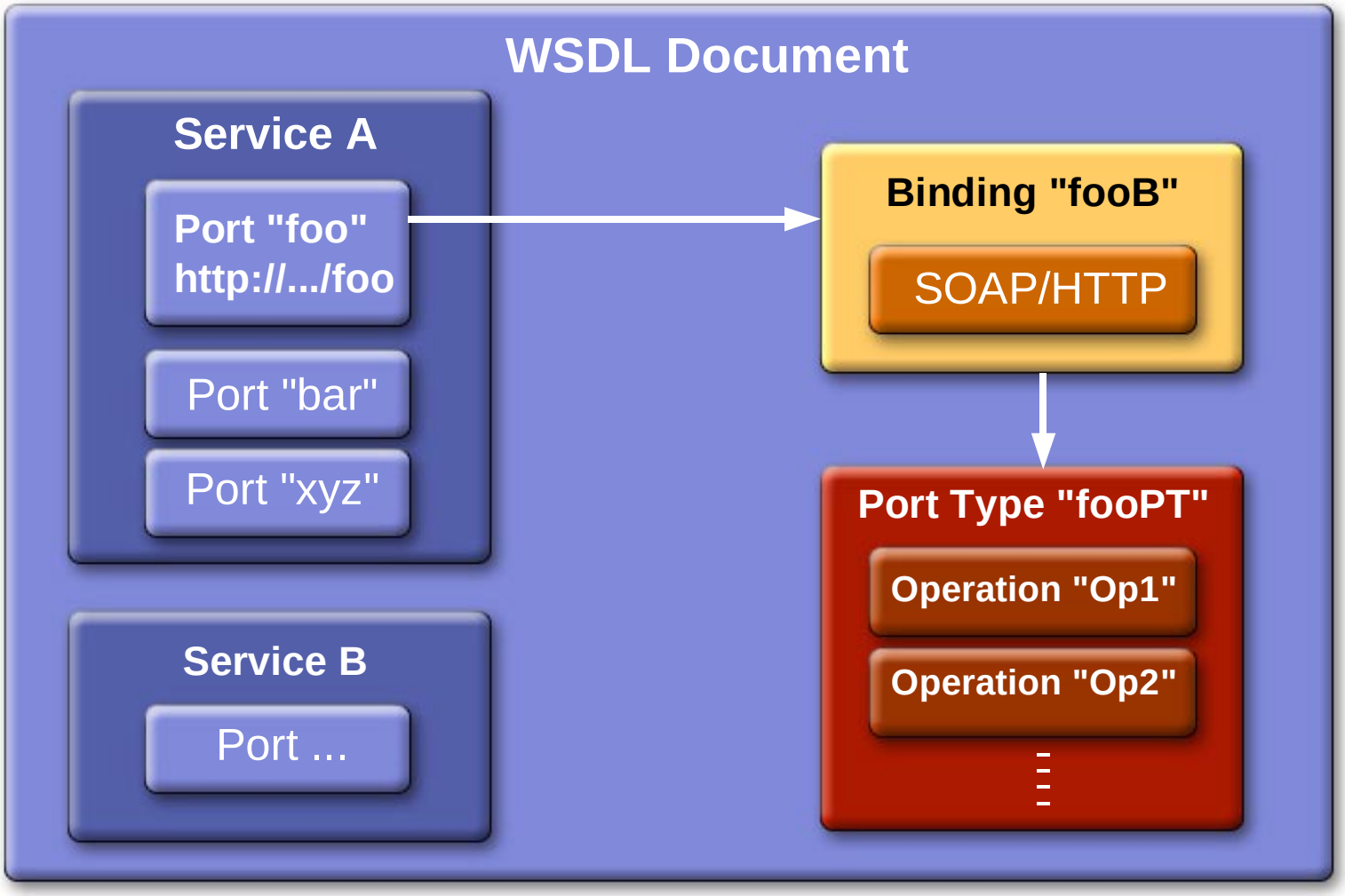
Brief Overview: SOAP

- XML-based protocol for exchange of information in a decentralized, distributed environment
- SOAP envelope
- Transport binding framework for exchanging messages using an underlying protocol
- Encoding rules for expressing instances of application-defined data types
- Convention for representing RPC requests and responses

Inside a SOAP Message



WSDL View of a Web Service



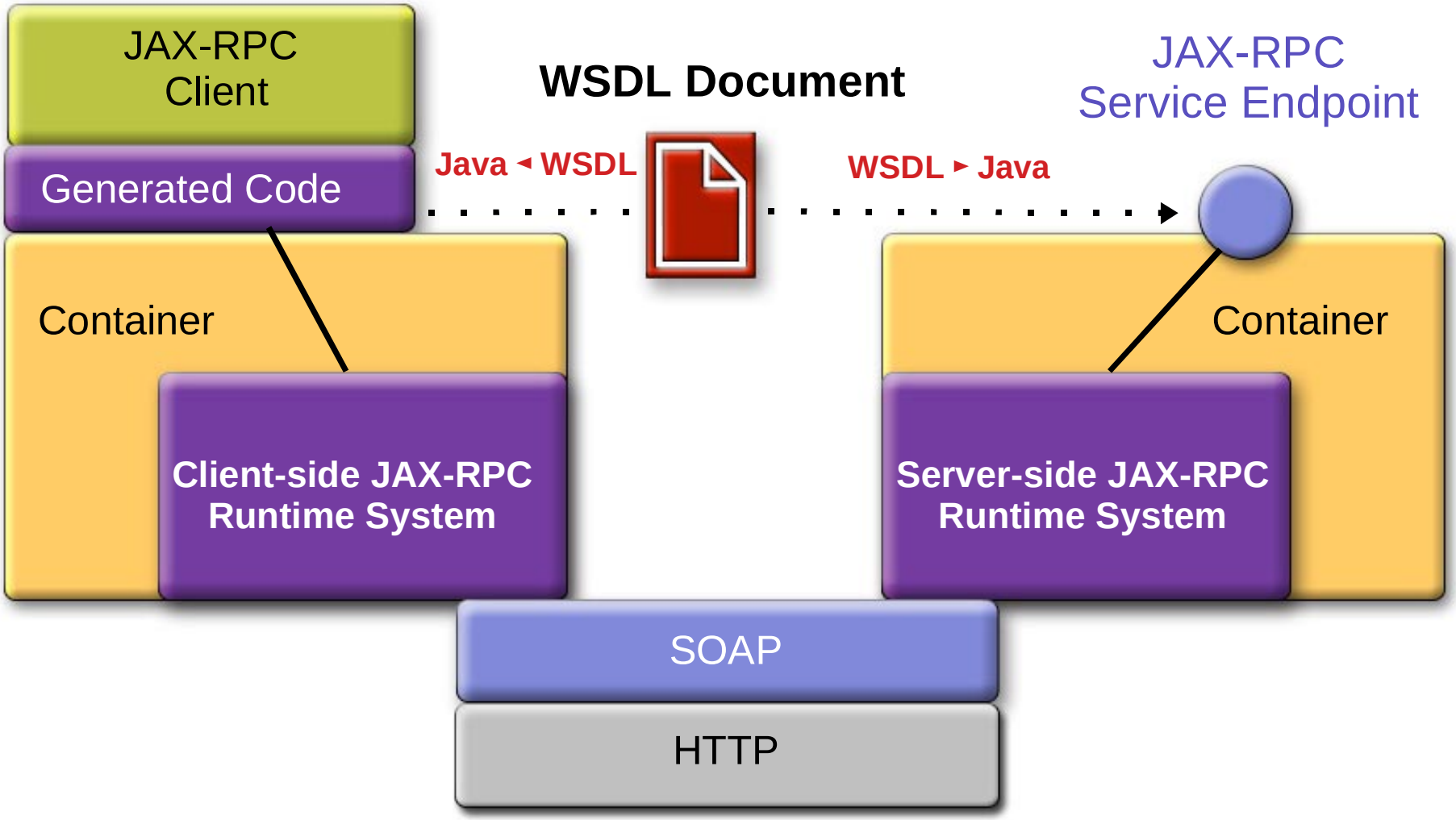
JAX-RPC Relationship to WSDL

JAX-RPC describes a Web Service as a collection of remote interfaces and methods

Tools are used to convert between WSDL documents and sets of Java™ remote interfaces

WSDL describes a Web Service as a collection of ports and operations

JAX-RPC Architecture Diagram

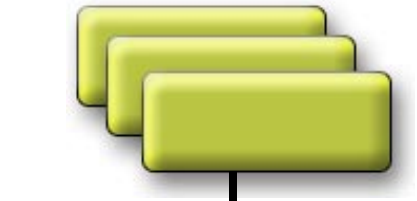


Sun microsystems

Push your development further

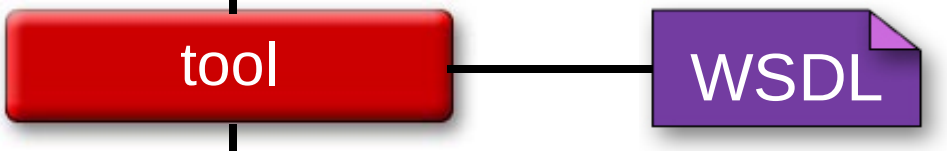
Developing a JAX-RPC-based Web Service

Developing a Web Service (Bottom-up approach)

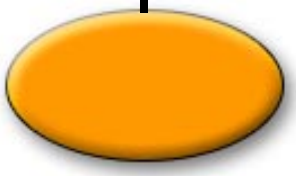


Interfaces (java.rmi.Remote)
Must follow JAX-RPC conventions

Implementation classes
Servlet-based endpoint model
Optional handler classes



Service contract



Packaged application (war/ear file)

Example: Interface

```
package hello;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface HelloIF extends Remote {
    public String sayHello(String s) throws RemoteException;
}
```

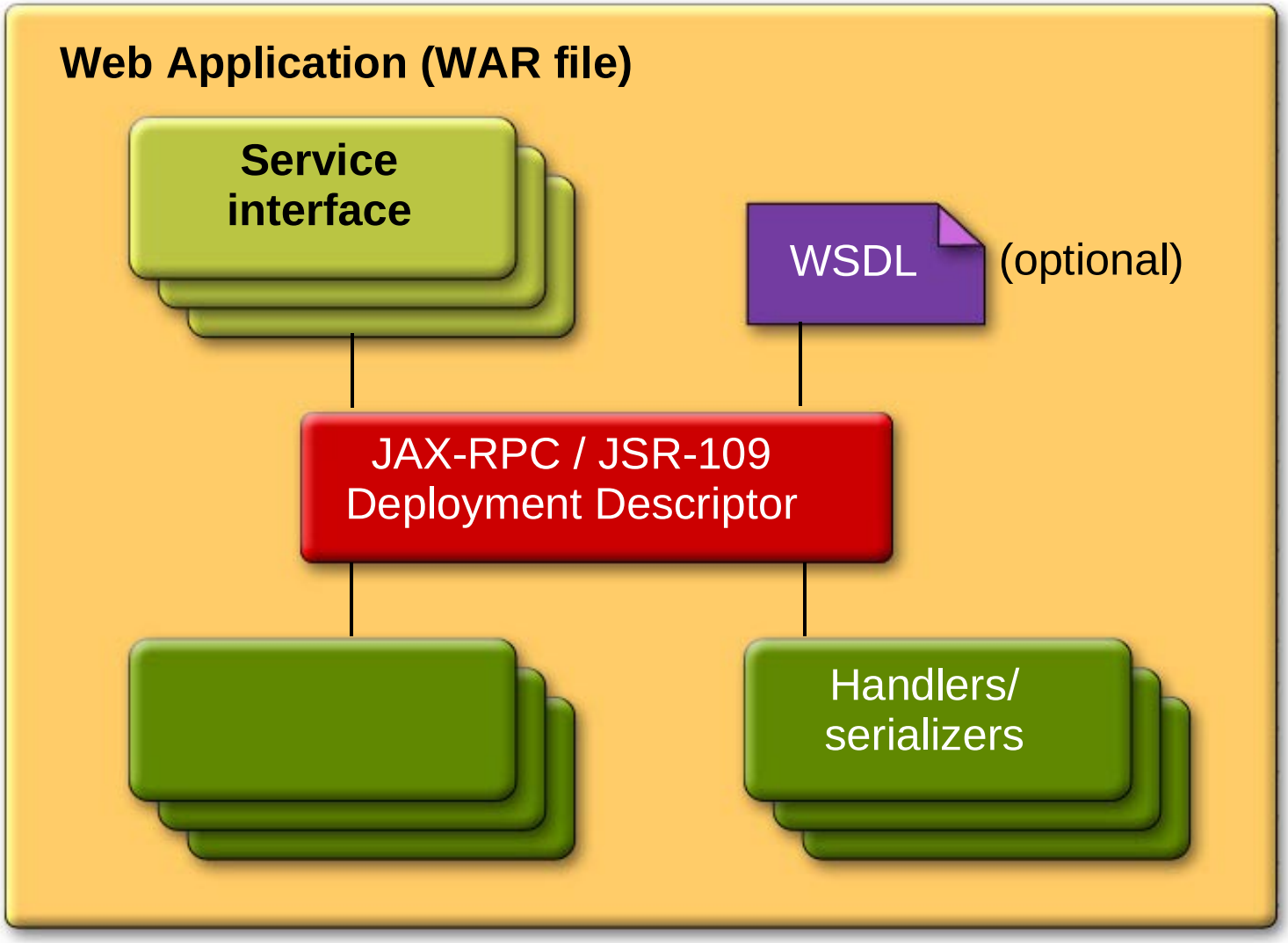
Example: Implementation

```
package hello;

public class HelloImpl implements HelloIF {

    public String message = new String("Hello ");

    public String sayHello(String s) {
        return new String(message + s);
    }

}
```

Packaging and Deployment

- Extension of Web-tier (WAR) or EJB-tier (EJB Module) packaging
- JSR-109 specifies standard deployment descriptor under J2EE 1.4
- Actual runtime model is container-specific
- Tools simplify packaging and deployment: Ant scripts, GUI wizards, IDE plug-ins

Packaging of JAX-RPC API-Based Applications



Demo

Building, Deploying, and Accessing a Web Service

Sun microsystems

Push your development further

Demo Scenario 1

- Exposing methods of a Java class as a Web service using Sun ONE Studio 5
- Packaging and deploying a Web service at Web-tier over Sun ONE App server
- Testing the Web service through a browser using automatically generated JSP test code by Sun ONE Studio 5

Demo Scenario 2

- Accessing Amazon.com Web service in real-time through a browser
- Accessing Amazon.com Web service in real-time using Swing application

JAX-RPC Client Programming Model

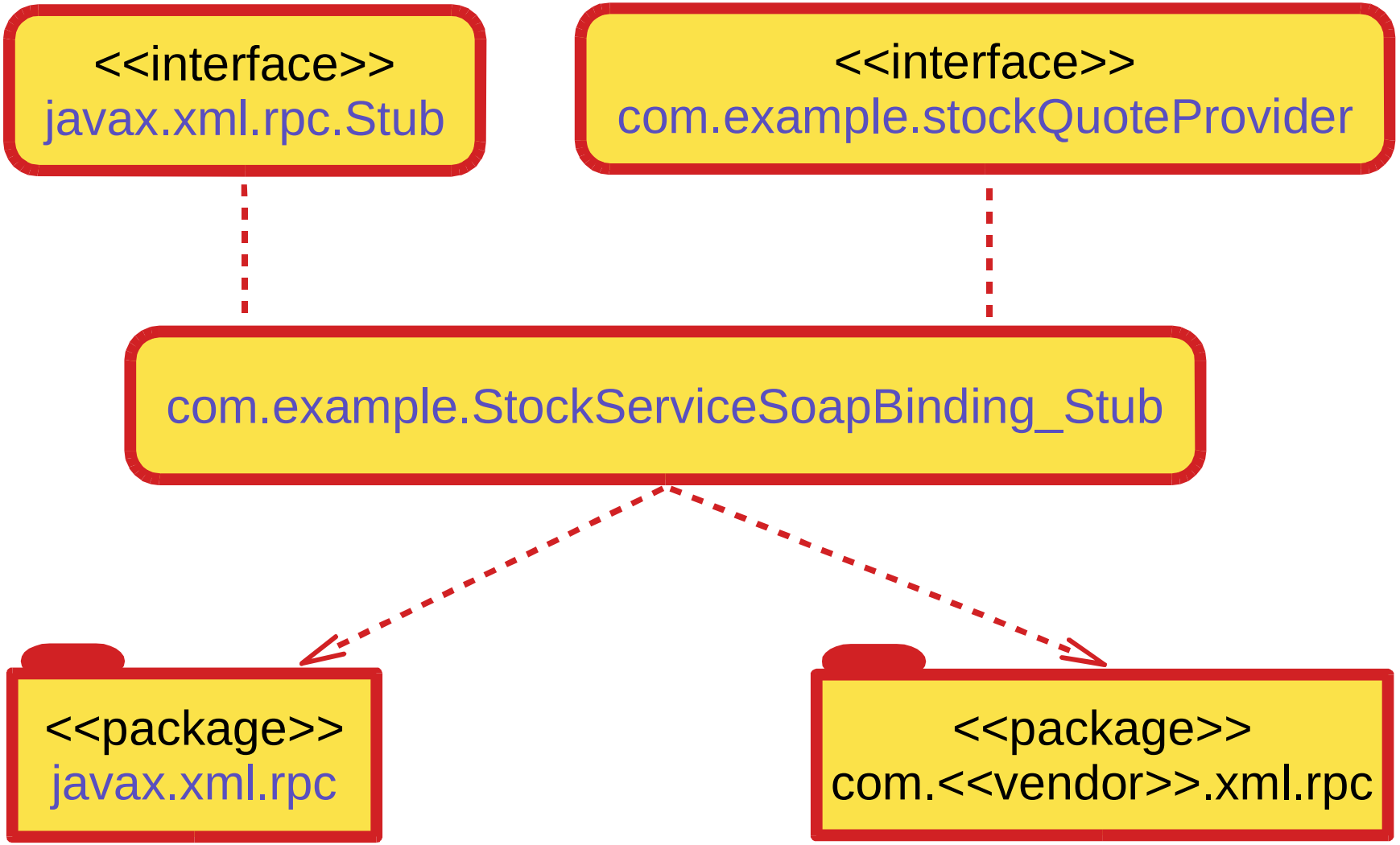
Client Programming Models

- Stub-based (least dynamic)
  - Both interface (WSDL) and implementation (stub) created at compile time
- Dynamic proxy
  - Interface (WSDL) created at compile time
  - Implementation (dynamic proxy) created at runtime
- Dynamic invocation interface (DII)
  - Both interface (WSDL) and implementation created at runtime

Stub-based Invocation Model

- Stub class gets generated at compile time
- All needed value classes are also generated
- Instantiated using vendor-generated Service implementation class
- Stub class is bound to a specific XML protocol (i.e. SOAP) and transport (i.e. HTTP)
- Best performance
- Stub class implements
  - Javax.xml.rpc.Stub interface
  - Web service definition interface

Stub Class Hierarchy



Example: Stub-based Client


```
package hello;

public class HelloClient {

public static void main(String[] args) {
  try {

    HelloIF_Stub stub = (HelloIF_Stub)
        (new HelloWorldService_Impl().getHelloIFPort());
    stub._setProperty(
        javax.xml.rpc.Stub.ENDPOINT_ADDRESS_PROPERTY,
        System.getProperty("endpoint"));

    System.out.println(stub.sayHelloBack("JAXRPC Sample"));
  } catch (Exception ex) {
    ex.printStackTrace();
  }
}

}
```

Dynamic Proxy-based Invocation Model

- Dynamic proxy is generated on the fly by JAX-RPC client runtime
- Application provides the Web Service definition interface that the dynamic proxy conforms to during runtime
- Easiest to program but slower than stub-based
  - Implementation object created and casted

DII Invocation Model

- Gives complete control to client programmer
- Most dynamic but complex programming
- Enables broker model
  - Client finds (through some search criteria) and invokes a service during runtime through a broker
  - Used when service definition interface is not known until runtime
  - You set operation and parameters during runtime
- Has to create Call object first

Example: DII Client

```
package dynamic;

import javax.xml.rpc.Call;
import javax.xml.rpc.Service;
import javax.xml.rpc.JAXRPCException;
import javax.xml.namespace.QName;
import javax.xml.rpc.ServiceFactory;
import javax.xml.rpc.ParameterMode;

public class HelloClient {

    private static String endpoint =
        "http://localhost:8080/dynamic-jaxrpc/dynamic ";
    private static String qnameService = "Hello";
    private static String qnamePort = "HelloIF";

    private static String BODY_NAMESPACE_VALUE =
        "http://dynamic.org/wsdl";
    private static String ENCODING_STYLE_PROPERTY =
        "javax.xml.rpc.encodingstyle.namespace.uri";
    private static String NS_XSD =
        "http://www.w3.org/2001/XMLSchema";
    private static String URI_ENCODING =
         "http://schemas.xmlsoap.org/soap/encoding/";
```

Example: DII Client

```
public static void main(String[] args) {
        try {
            ServiceFactory factory = ServiceFactory.newInstance();
            Service service =  factory.createService(new QName(qnameService));
            QName port = new QName(qnamePort);

            Call call = service.createCall(port);
            call.setTargetEndpointAddress(endpoint);

            call.setProperty(Call.SOAPACTION_USE_PROPERTY, new Boolean(true));
            call.setProperty(Call.SOAPACTION_URI_PROPERTY,"");
            call.setProperty(ENCODING_STYLE_PROPERTY, URI_ENCODING);
            QName QNAME_TYPE_STRING = new QName(NS_XSD, "string");
            call.setReturnType(QNAME_TYPE_STRING);
            call.setOperationName(new QName(BODY_NAMESPACE_VALUE "sayHello"));
            call.addParameter("String_1", QNAME_TYPE_STRING, ParameterMode.IN);
            String[] params = { "Duke!" };

            String result = (String)call.invoke(params);
             System.out.println(result);

        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}
```

Demo

JAX-RPC Client Programming Model

Push your development further

Demo Scenario

- Building and running 3 different client programming models using Sun ONE Studio 5
  - Stub, Dynamic proxy, DII
- Measuring the duration of the call under each programming model

SOAP Message Handler

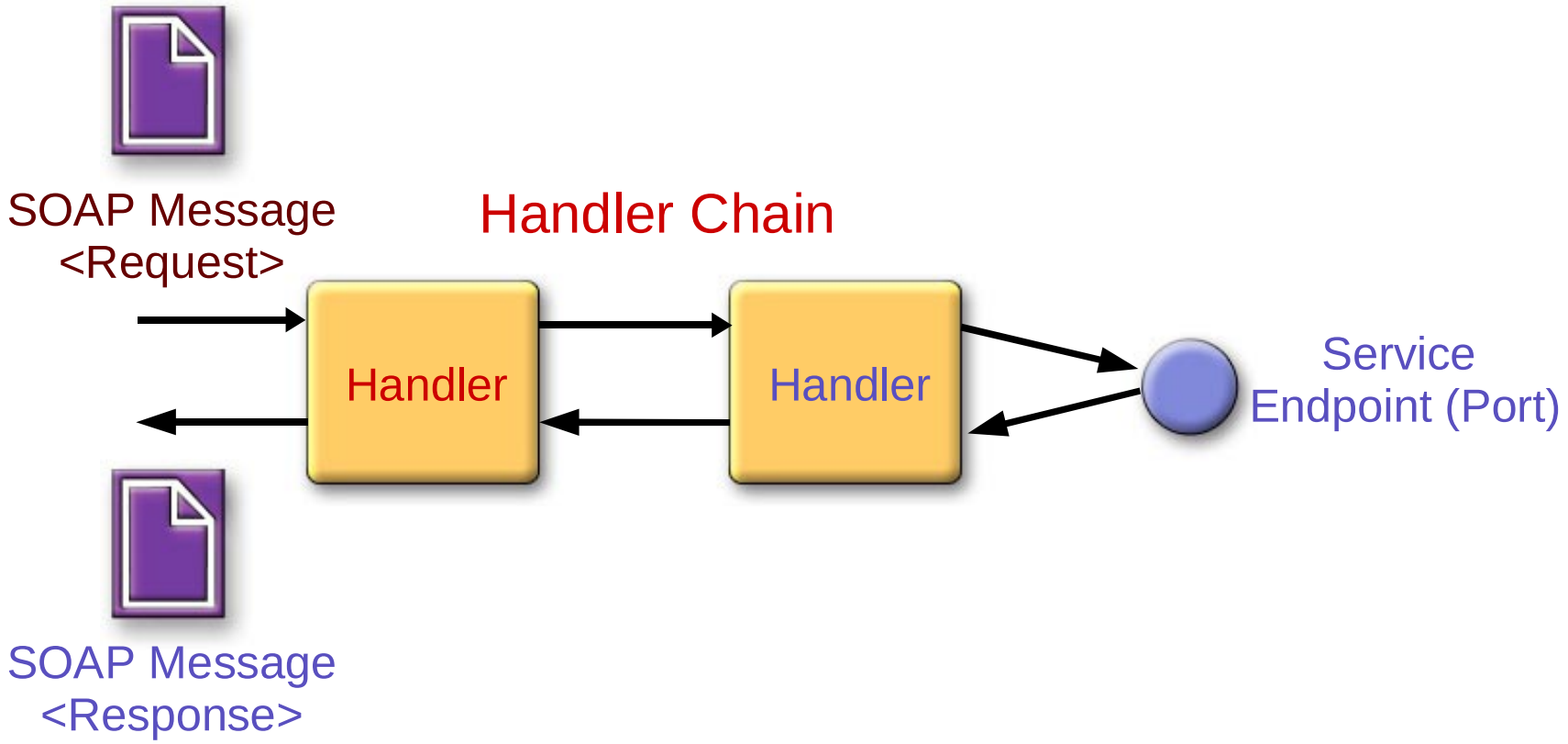
SOAP Message Handlers

- Handlers let you access/modify SOAP request and response messages
  - Typically used to process service contexts in SOAP header blocks
  - Can be used to extend functionality of Web Services runtime system
    - J2EE containers (which provide Web Services runtime) are likely to use them internally to provide session/transaction propagation
- Example handlers: Encryption, decryption, authentication, authorization, logging, auditing, caching

SOAP Message Handlers

- Pluggable and chainable
  - Through standardized programming API
  - Portable across implementations
- Has its own life-cycle
  - JAX-RPC runtime system calls init(), destroy() of a handler
- Handler instances can be pooled
- MessageContext is used to share properties among handlers in a handler chain

SOAP Message Handlers



Example SOAP Message Handler

```
package com.example;

public class MySOAPMessageHandler implements
                                    javax.xml.rpc.handler.Handler {
  public MySOAPMessageHandler() { ... }
  public boolean handleRequest(MessageContext context,
                                    HandlerChain chain){
    try {
      SOAPMessageContext smc = (SOAPMessageContext)context;
      SOAPMessage msg = smc.getMessage();
      SOAPPart sp = msg.getSOAPPart();
      SOAPEnvelope se = sp.getEnvelope();
      SOAPHeader sh = se.getHeader();
      // Process one or more header blocks
      // ...
      // Next step based on the processing model for this handler
    }
    catch(Exception ex) {
      // throw exception
    }
  }
  // Other methods: handleResponse(), handleFault(), init(),
   destroy()
}
```

Demo

SOAP Message Handler

Demo Scenario

- Building and configuring two server side SOAP message handlers as a chain using Sun ONE Studio 5
- Building and programmatically deploying client side SOAP message handler

JAX-RPC Runtime Services

Push your development further

Session Management

- JAX-RPC runtime system manages session
  - Service client or service developer do not have to deal with session management
- Supported session management schemes over HTTP
  - Cookie-based
  - URL rewriting
- SOAP header-based session management scheme in the future

Security

- HTTP basic authentication support is required
- Certificate based mutual authentication using HTTP/S (HTTP over SSL)
  - J2EE 1.4 mandates it
- Does not require support for SOAP Security Extensions for digital signature
  - J2EE 1.4 does not mandate it but vendor products will support it (Java WSDP 1.2 supports it now, Sun ONE App Server 8 will support it in the future)

Example: HTTP Basic Authentication

```
StockQuoteService sqs = getStockQuoteService(..);

// Get the instance of stub object setting username & password
StockQuoteProvider sqp = sqs.getStockQuoteProviderPort(
                    "<username>",
                    "<password>");
float quote =
    sqp.getLastTradePrice("ACME");
```

Sun microsystems

Push your development further

Demo

Basic Authentication

Demo Scenario

- Redeploying a Web service with Basic authentication enabled
- Running client application without passing username and password - it should fail with “authorization failure”
- Running client application with username and password

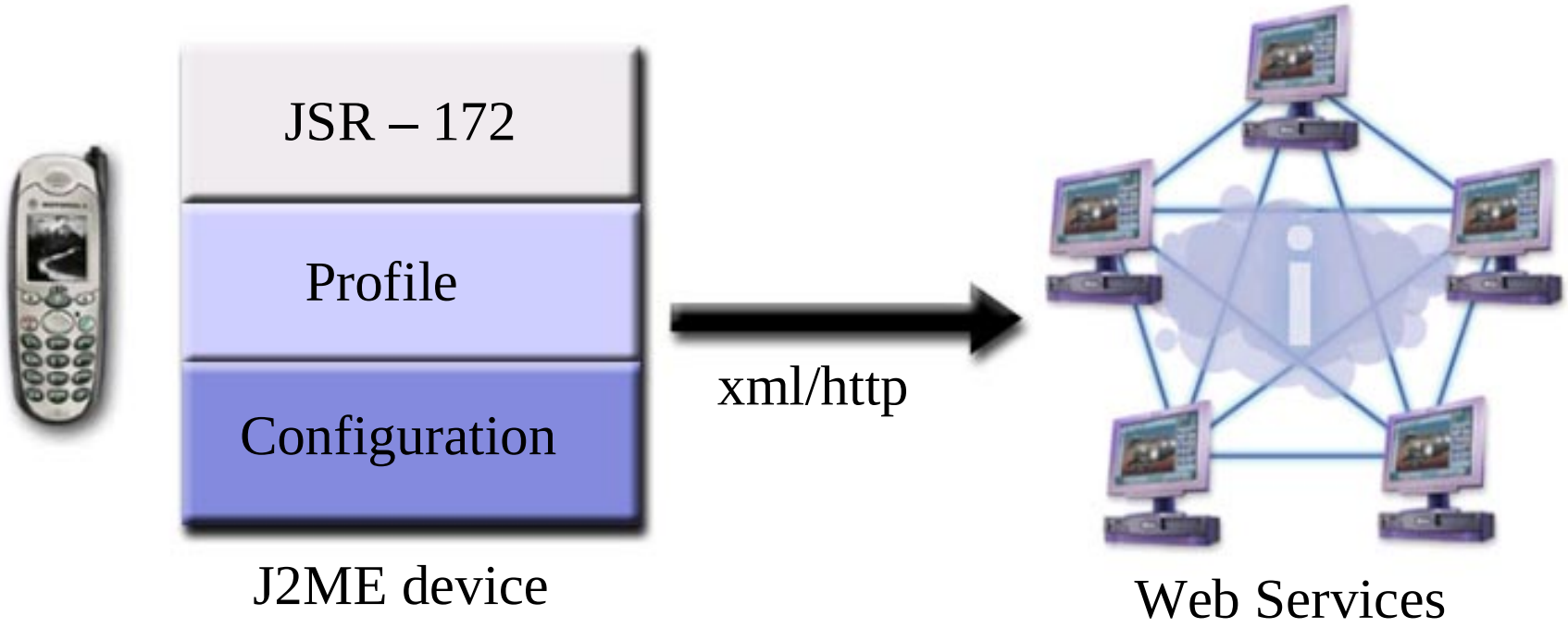
J2ME and Web Services

Sun microsystems

Push your development further

JSR-172 (J2ME Web Services)

- Parsing
- J2ME Web services client



JAX-RPC Subset of JSR172

- Subset of JAX-RPC 1.0
- Additionally specifies runtime SPI-portable stubs
- No support for the service endpoint model. The subset only provides support for clients to access web service endpoints.
- Alignment with WS-I Basic Profile
- Protocol encoding: SOAP 1.1 using XML based protocol

Sun microsystems

Push your development further

Demo

J2ME Web Service

Demo Scenario

- Building and running J2ME Web service client application (through an emulator) using Sun ONE Studio 5

Document-style Web Services

Sun microsystems

Push your development further

RPC vs. Document-driven

Sun™ Tech Days

Sun microsystems

RPC

- Procedure call
- Method signature
- Marshalling
- Tightly-coupled
- Point to point
- Synchronous
- Typically within Intranet

Document-driven

- Business documents
- Schema
- Parsing & Validating
- Loosely coupled
- End to end
- Asynchronous
- Typically over internet

When to use RPC and When to use Document-driven?

Sun™ Tech Days

Sun microsystems

RPC

- Within Enterprise
- Reliable and high bandwidth
- Short running business process
- Trusted environment

Document-driven

- Between enterprise and enterprise
- Unpredictable bandwidth
- Long running business process
- Blind trust

Document-Driven Model using JAX-RPC

- Use of “document/literal” SOAP message (instead of “RPC/encoding”)
  - SOAP body contains XML document, i.e. Purchase order
  - Specified via “style” and “use” attribute in WSDL document
- Use of Attachments
  - Attachment contains XML document
  - Specified via MIME binding in WSDL document

Document-Style WSDL

- WSDL provides a document-style service contract between sender and receiver
- Abstract Message Description
  - Provides name for each part: PARTNAME
  - Provides type of each message part (e.g., schema for XML parts)
- Binding Description
  - Provides messaging packaging format

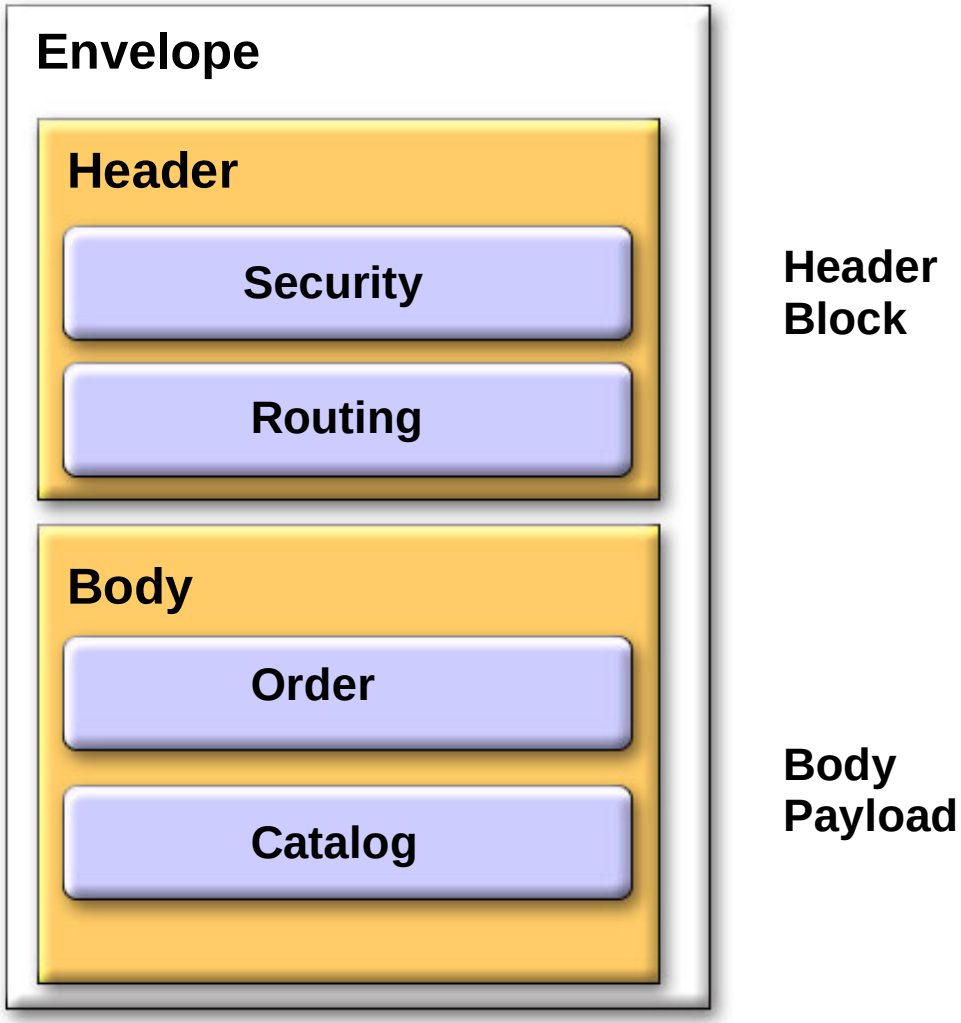
Document-Style SOAP

Header

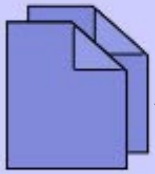
- Specifies message-level services

Payload

- Opaque
- Schema-defined
- Large
- Complex

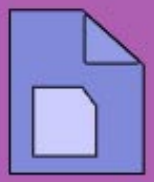


SOAP Attachments and Literals



Attachments

- Very non-restrictive
  - XML
    javax.xml.transform.Source
  - Non-XML
    javax.activation.DataHandler
- Parse-your-own
  or use JAXB



Literals

- XML doc/structure embedded in SOAP body
- JAX-RPC parses XML, creates java objects
- Obtains service information from WSDL
- Several XSD elements optional - may not be implemented by some appservers

WS-I & Web Services Interoperability

WS-I Is

"An open industry effort chartered to promote Web Services interoperability across platforms, applications, and programming languages.

The organization brings together a diverse community of Web services leaders to respond to customer needs by providing guidance, recommended practices, and supporting resources for developing interoperable Web services."

WS-I Is Not

- Is not a source of WS-* specs
  - These have typically been proprietary specifications from single or small groups of companies, though a few have been submitted to recognized standards organizations
- Is not a ‘standards’ organization
  - Doesn’t produce specs for new technology
  - Profiles existing specifications

Basic Profile 1.0

- Profiling
  - SOAP 1.1, WSDL 1.1 and UDDI 2.0
- Consists of 156 conformance requirement
  - 48 related to SOAP
  - 84 related to WSDL
  - 8 related to UDDI
  - 6 related to security

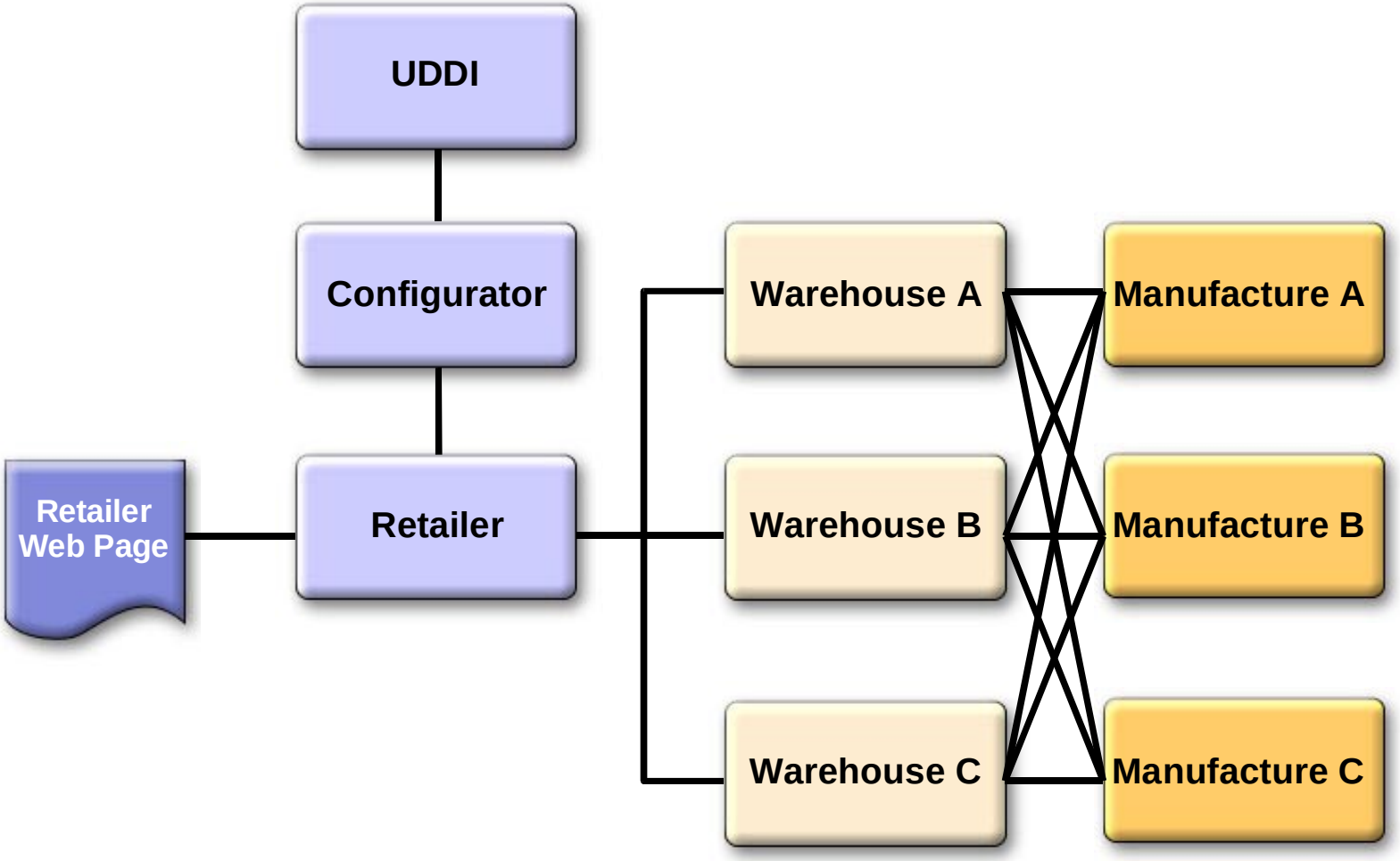
Basic Profile 1.1

- Adds support for attachments to Basic Profile 1.0
- Based on
  - SOAP+Attachments W3C Note (MIME)
  - WSDL MIME Binding
- Currently an editors draft
- Expected to become final in October

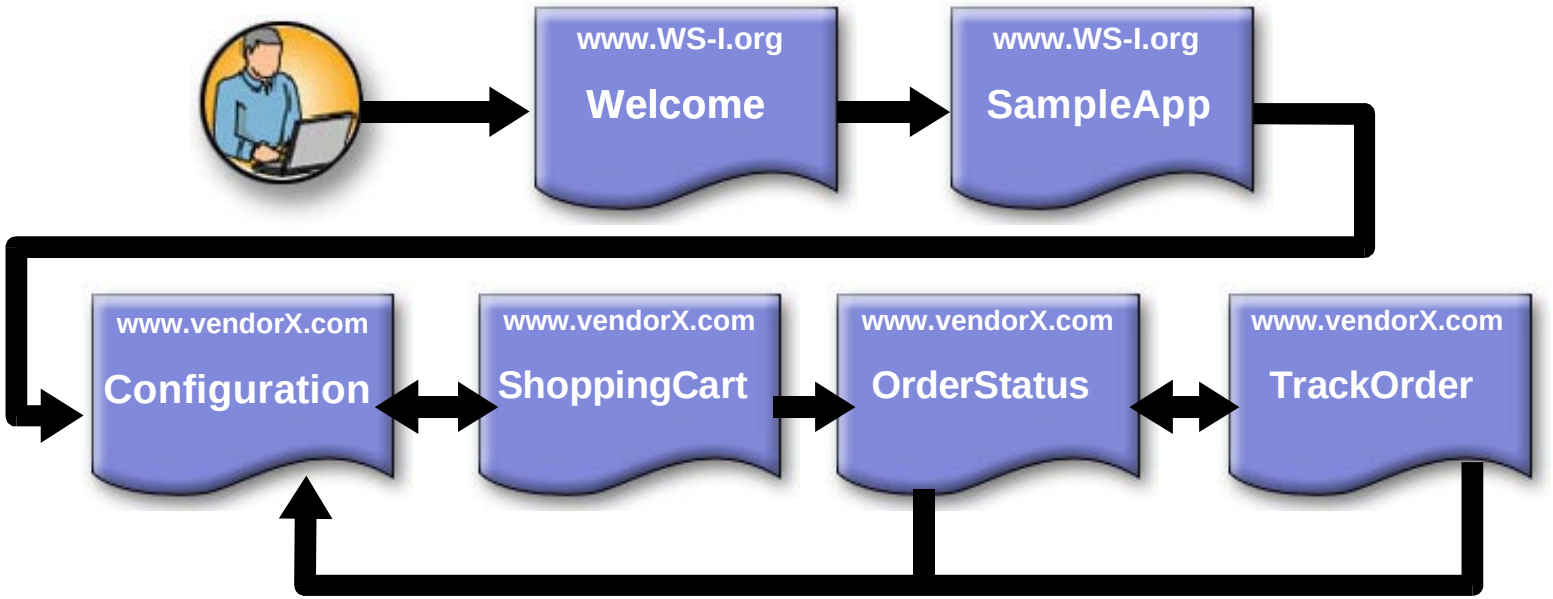
WS-I Support in J2EE 1.4

- Package WS-I BP 1.0-conforming WSDL documents in your J2EE™ 1.4 application
- Containers will take care of all the details:
  - HTTP 1.1 requirements
  - SOAP 1.1 requirements
  - WSDL 1.1 requirements
  - UDDI 2.0 requirements (if supported)

Supply Chain Management Sample Application



Sample Application Flow



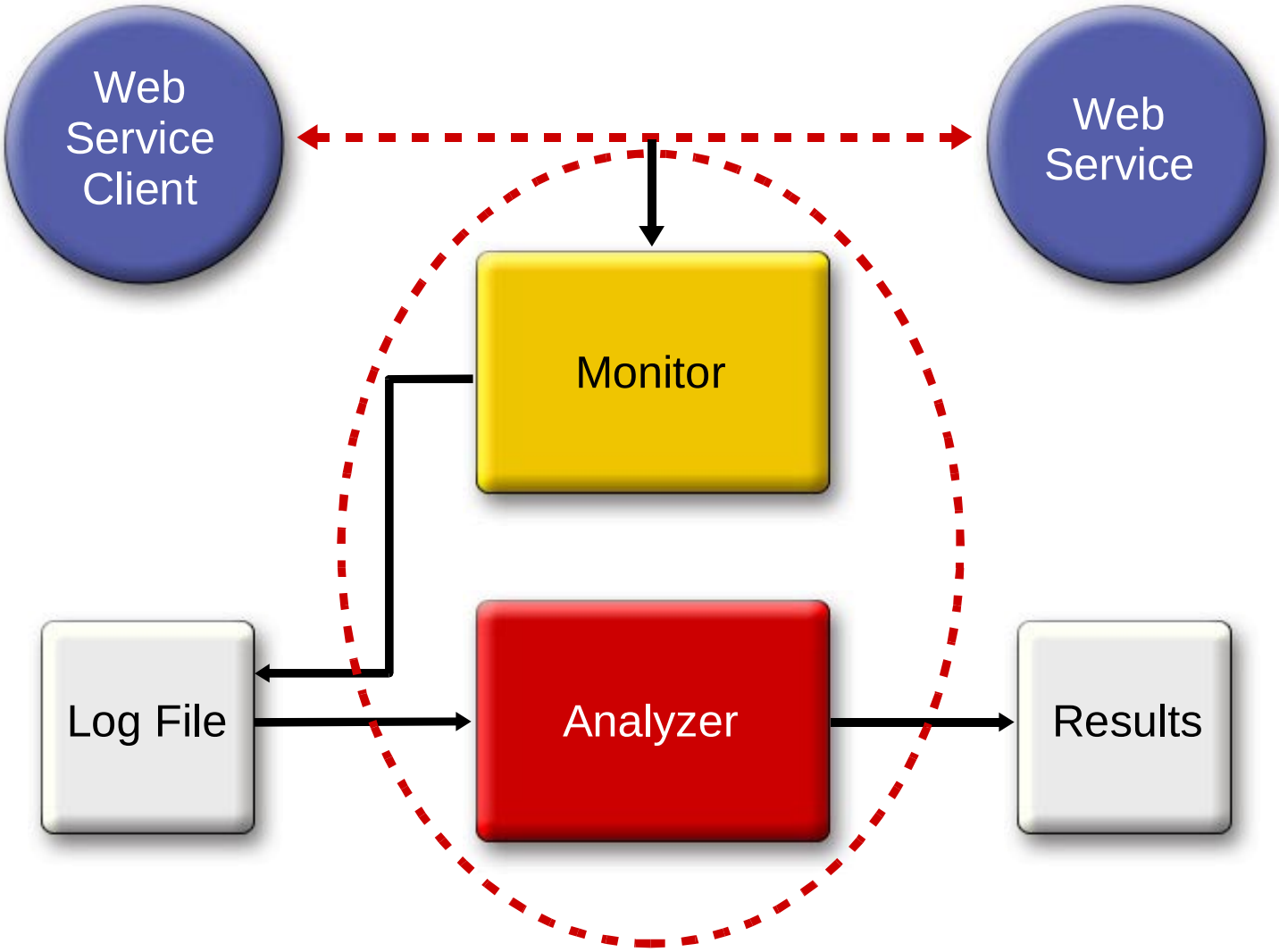
WSDL Descriptions

Sample Application Supply Chain Management Architecture - StarOffice 6.0

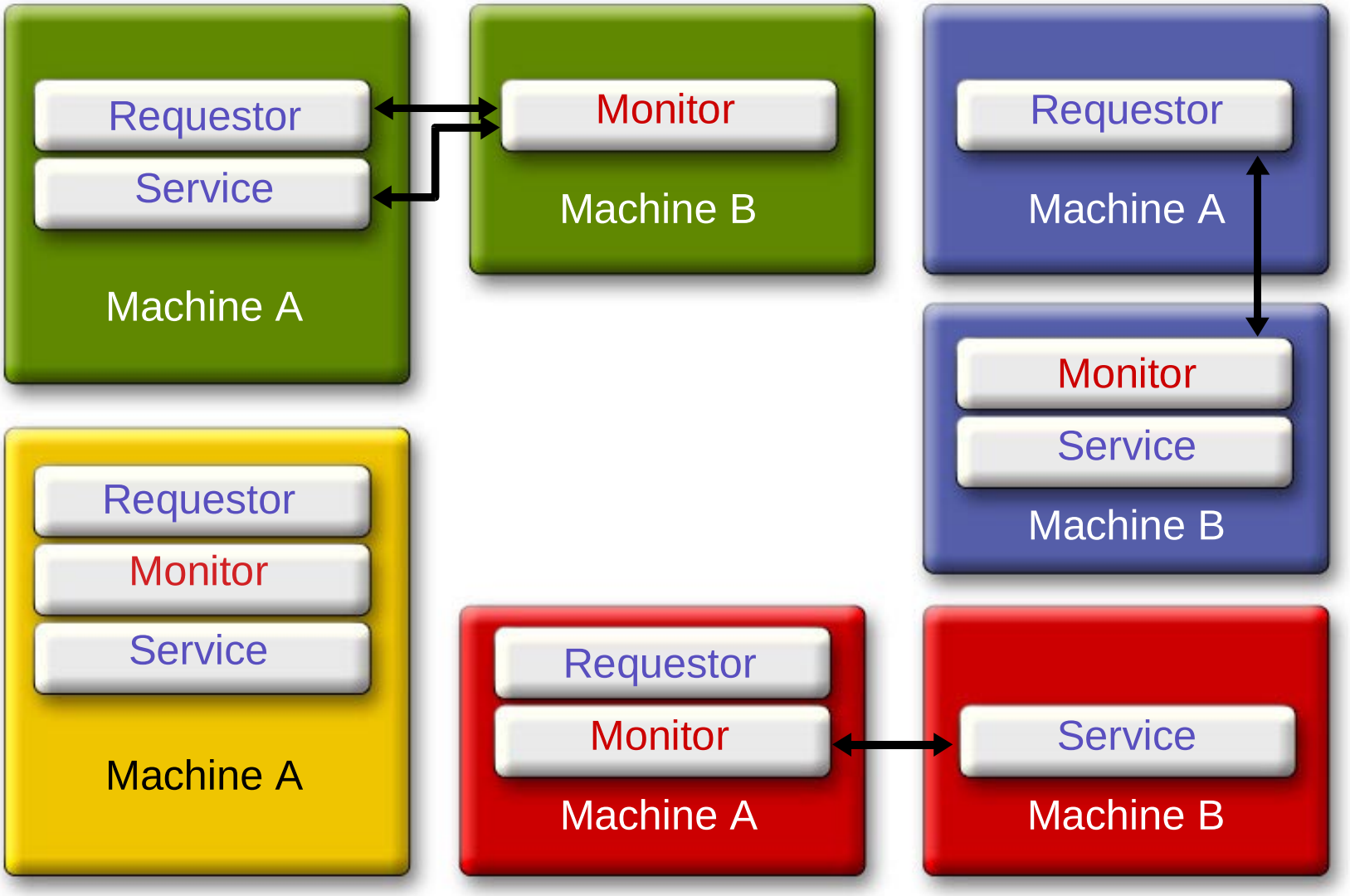
File Edit View Insert Format Tools Window Help

	Retailer	Warehouse	WH Callback	Manufacturer	Logging Facility	Configurator
Usage Scenario	Request-ResponseRequest/ Response	Request-ResponseRequest/ Response	Basic Callback; Request-ResponseRequest/ Response	Basic Callback; Request-ResponseRequest/ Response	One WayOne-way; Request-ResponseRequest/ Response	Request/- Response
Style	Rpc/literal	Rpc/literal	Doc/literal	Doc/literal	Doc/literal	Doc/literal
Header	Yes	Yes	Yes	Yes	No	No
Data types	nonnegative Integer decimal string integer normalizedString NMTOKEN anyURI	normalizedString nonnegative Integer unsignedShort boolean NMTOKEN anyURI string	nonnegative Integer unsignedShort float string normalizedString NMTOKEN anyURI	string float normalizedString nonnegative Integer unsignedShort NMTOKEN	string dateTime NMTOKEN	boolean string NMTOKEN anyURI
Attribute	Yes	Yes	Yes	Yes	No	Yes
SOAP Action	Empty string	WSDL tns	None	None	Empty string	WSDL tns + operation
Naming	Mixed	Start with upper case	Mixed	Mixed	Start with upper case	Start with lower case

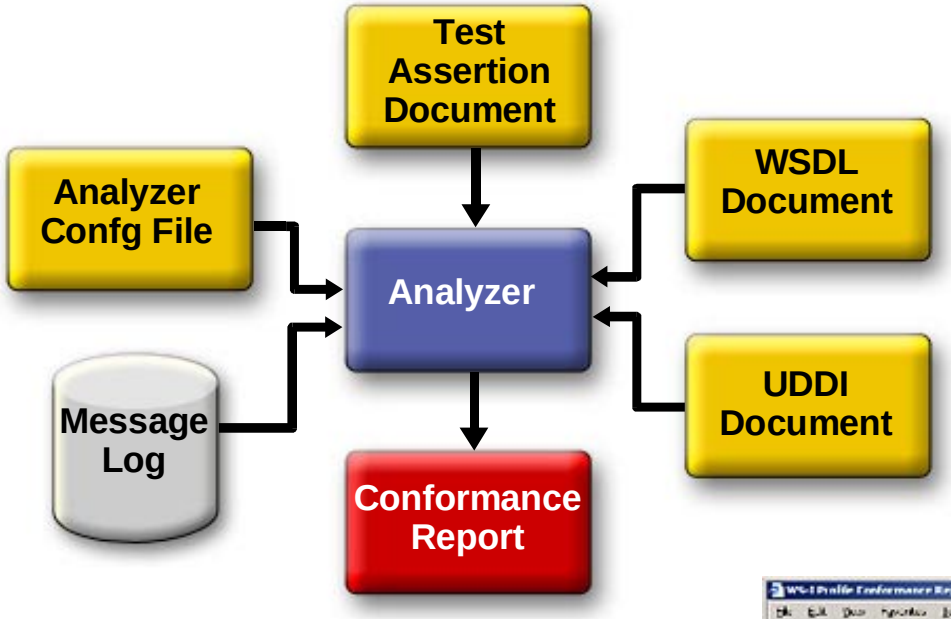
Monitor & Analyzer

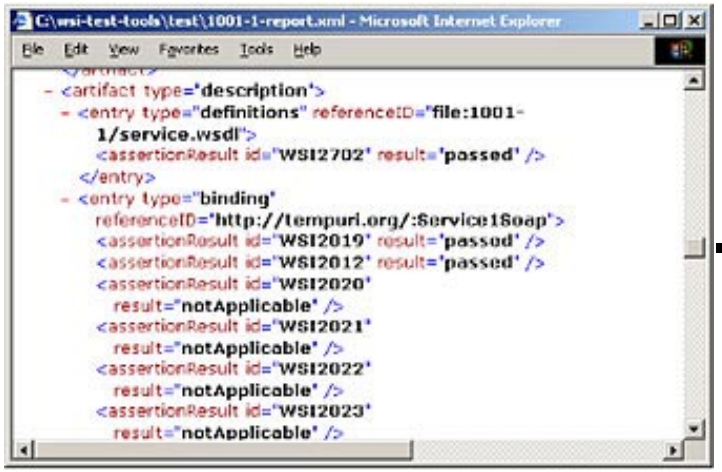


Monitor Can Live Anywhere



Analyzer


Test Assertion Document
Analyzer Confg File
WSDL Document
Analyzer
Message Log
UDDI Document
Conformance Report


C:\wsi-test-tools\test\1001-1-report.xml - Microsoft Internet Explorer
File Edit View Favorites Tools Help
- <artifact type="description">
- <entry type="definitions" referenceID="file:1001-1/service.wsdl">
<assertionResult id="WSI2702" result="passed" />
</entry>
- <entry type="binding" referenceID="http://tempuri.org/:Service1Soap">
<assertionResult id="WSI2019" result="passed" />
<assertionResult id="WSI2012" result="passed" />
<assertionResult id="WSI2020" result="notApplicable" />
<assertionResult id="WSI2021" result="notApplicable" />
<assertionResult id="WSI2022" result="notApplicable" />
<assertionResult id="WSI2023" result="notApplicable" />
XSLT
Summary
Result passed
Artifacts
discovery
description
message
Artifact: discovery
Assertion Result Summary:
Assertion ID | Passed | Failed | Warning | Not Applicable | Not Testable
WSI3002 | 0 | 0 | 0 | 0 | X
WSI3003 | 0 | 0 | 0 | 0 | X

Sun microsystems

Push your development further

Demo

WS-I Sample Application

Demo Scenario

- Running WS-I Supply Chain Management sample application accessing service endpoints from various companies over the internet
  - Sun, IBM, BEA, Oracle
- Running Monitor and Analyzer

Changing Landscape of Web Services

Sun microsystems

Push your development further

New Web Services Technologies

- ebXML
- Fast Web service
- Metadata Web service
- Orchestration
- Transaction
- Reliable messaging
- Security

ebXML

ebXML Vision

A global electronic market place where enterprises of any size, anywhere can:

- Find each other electronically
- Conduct business through exchange of XML based business messages

Why ebXML?

- Automation of business collaboration
- Need for standardizing business collaboration
  - What are the business processes?
  - Who are the parties involved in business collaboration? What are their roles?
  - What and how do XML documents get exchanged in the business collaborations?
  - What are the security, reliability, quality of service requirements of this business collaboration?

Why ebXML?

SOAP, WSDL, UDDI alone are not adequate!

- WSDL does not address business collaboration
- SOAP (in its basic form) does not provide secure and reliable message delivery
- UDDI does not provide repository capability for business objects

Why ebXML?

Existing B2B Frameworks are not adequate!

- EDI
  - Too heavy-weight and too rigid
  - VPN
  - Customization for each B2B instance
- RosettaNet
  - PIP definitions are somewhat rigid and cannot be discovered per partner basis
- BizTalk
  - Proprietary, Single-vendor, Single-platform
  - No concept of Business collaboration & Partner profile

Web Services Adoption Phases

- 1st Phase – Simple Web Services (Now)
  - Consumer-focused, stateless
- 2nd Phase – EAI Web Services (Begun)
  - Deployed within organization boundaries to enable internal integration
- 3rd Phase – Business (B2B) Web Services (2004?)
  - Deployed on Extranets to enable business process integration with trading partners, customers, other players in your value chain

Simple Web Services (WUS) vs. Business Web Services (ebXML)

Sun™ Tech Days

Sun microsystems

Simple Web Services

- Simple interaction
- Consumer oriented
- Short-living process
- No business collaboration
- No partner profile
- Not secure, not reliable
- Does not support non-repudiation
- No repository support

Business Web Services

- Complex interaction
- Business oriented
- Long-running process
- Supports business collaboration
- Supports partner profile
- Secure and reliable and non-repudiation
- Supports non-repudiation
- Registry and repository

EAI vs. Business Web Service

EAI

- Within a business organization
- Centralized control
- Implicit contract
- Small number of business processes and participants

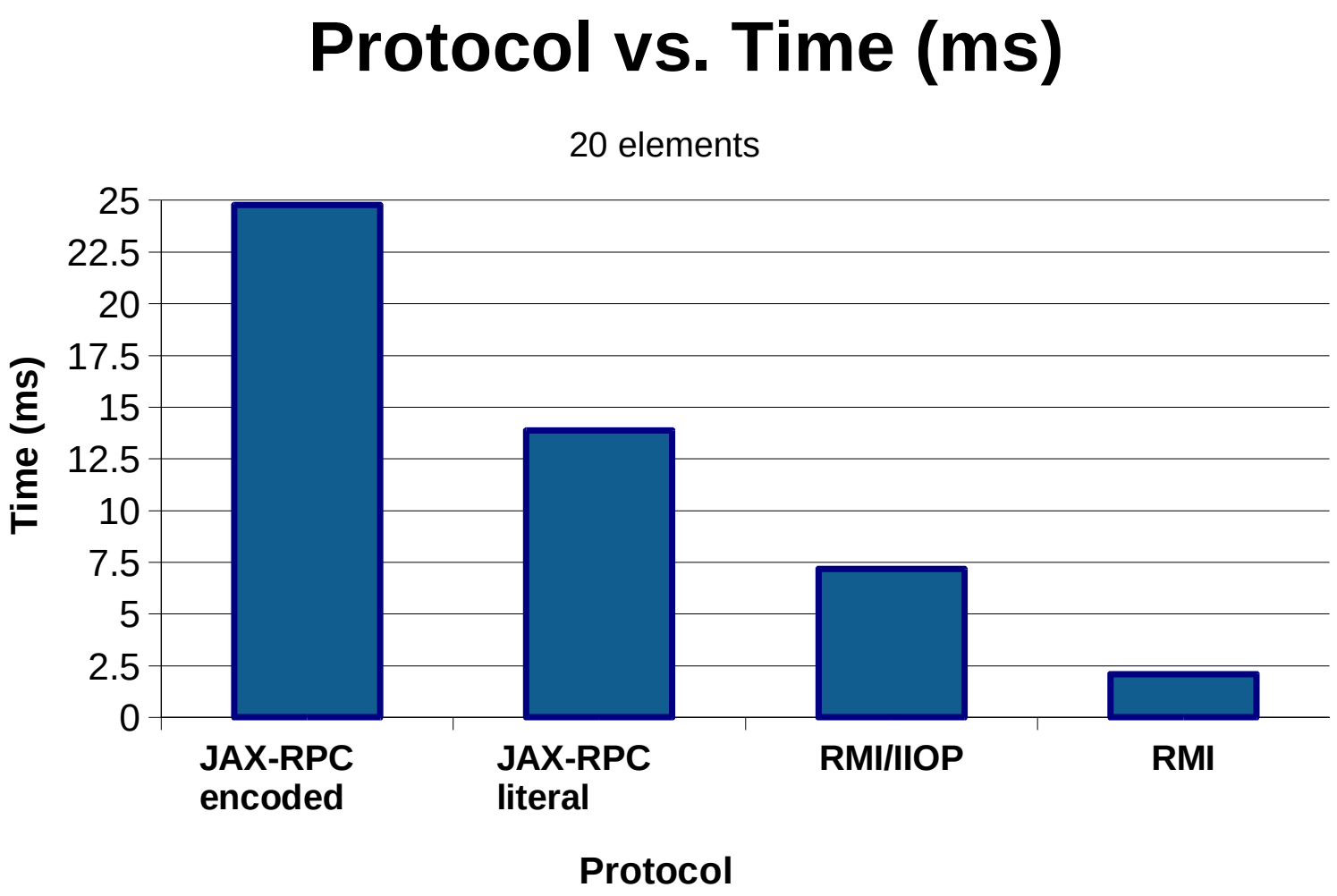
Business Web Service

- Between business organizations
- Distributed control
- Explicit contract
- Potentially large number of business processes and participants

Fast Web Service

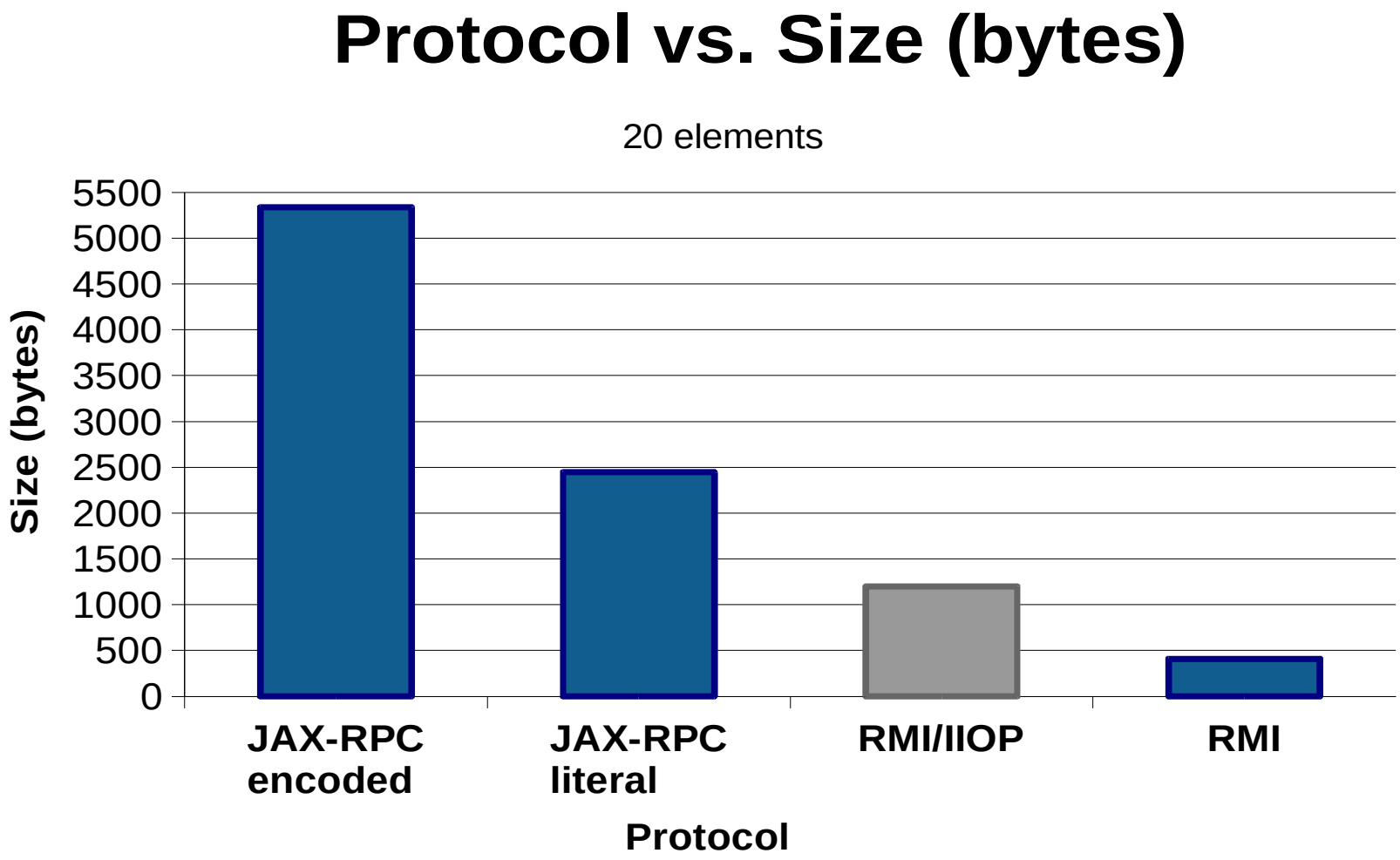
Current Performance Data

Loopback Request/Response Latency



Current Performance Data

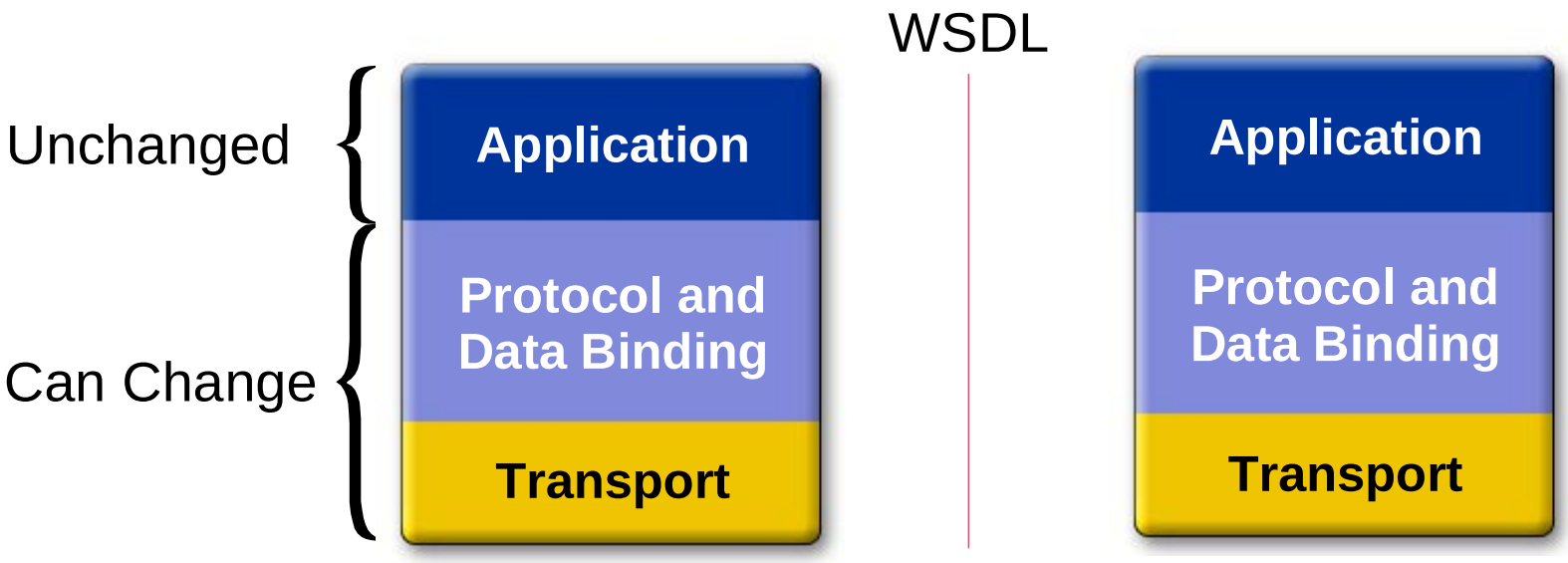
Message Size



Goals of Fast Web Services

- Provide much better performance
- Standards for Fast Web Services
  - Interoperability
- Take advantage of Java™ Web Services stack
  - Fast implementation in stack
- Minimize impact to Web Service developers
  - Runtime stack will hide the details

The Big Picture



Technical Goals

- Cut overhead of XML processing
  - SOAP message size
  - Marshaling to programmatic types
- Maximize use of APIs, tools and standards
  - JAX-* APIs, WSDL
- Support for J2ME™, J2SE™ and J2EE™ technologies
  - JSR-172, Web Services for J2ME™
  - End-to-end support
- Platform and programming language independent

Use Cases

- Web Services within the enterprise
- Time- and resource-sensitive systems
  - Mobile phones
  - Satellites
- High-performance computing
  - Grid computing
  - Scientific computing
- Auto-ID

Technological Requirements

- Consistent non-specific encoding technology
  - Fast infoset, Fast schema and Fast SOAP
  - Not specific to application
- Proven use in network communications
  - Large-scale deployment
- Platform and programming language independent
- Existing standards
  - Royalty-free and open

Abstract Syntax Notation One (ASN.1)

Sun™ Tech Days

Sun microsystems

- Schema language for abstract type system
- Multiple encoding rules
  - Types are independent of encoding
- Royalty-free set of standards at ITU-T/ISO
- In development for nearly 20 years
- Extensively used in telecom industry
- Implementations in Java™, C and C++ programming languages

Fast Encoding and ASN.1

- Fast infoset encoding
  - ASN.1 Schema for XML infoset
- Fast schema encoding
  - W3C XML Schema to ASN.1 mapping
- Fast SOAP encoding
  - ASN.1 Schema for SOAP
- Packed Encoding Rules (PER)
  - Most compact and CPU efficient
  - Other rules could be used (e.g., DER)

Demo

Fast Web Service

Sun microsystems

Push your development further

Demo Scenario

- Comparing regular Web service and fast Web service performance in real time using different size of the messages

Push your development further

Metadata-driven Web Service (JSR 181)

Goals

- Simplify Web services development and deployment dramatically
- Leverage Java Language Metadata technology (JSR 175)
  - provide an easy to use syntax for describing web services at the source-code level
- Use standard Java compiler (J2SE 1.5)
  - Validate Web services metadata
  - Produce class files containing metadata
- Allow Web services metadata to be manipulated by tools

Goals

- Enable auto-deployment
  - Like JSP deployment
- Abstract away details of Web Service implementation and deployment
  - Protocols, WSDL, service endpoints, XML/Java™ mapping, message formats, deployment descriptors, packaging
- Built over existing Web services APIs and technologies
  - Hide low-level programming APIs for Web services components and J2EE
  - Like JSP to Servlet

JSR 181

- Java™ Web Service (JSR 181 WS) file is central
  - Both source and compiled form
  - Web Service metadata annotates 181 WS file
  - 181 WS file is a standard Java™ source file
- JSR 175 used to represent metadata (J2SE 1.5)
  - A Java™ language extension with compiler support
  - Define Metadata vocabulary for application area
    - Web Services (JSR 181 defines vocabulary)
  - Metadata in class file and available at run-time

An Example (Part of a 181 WS File)

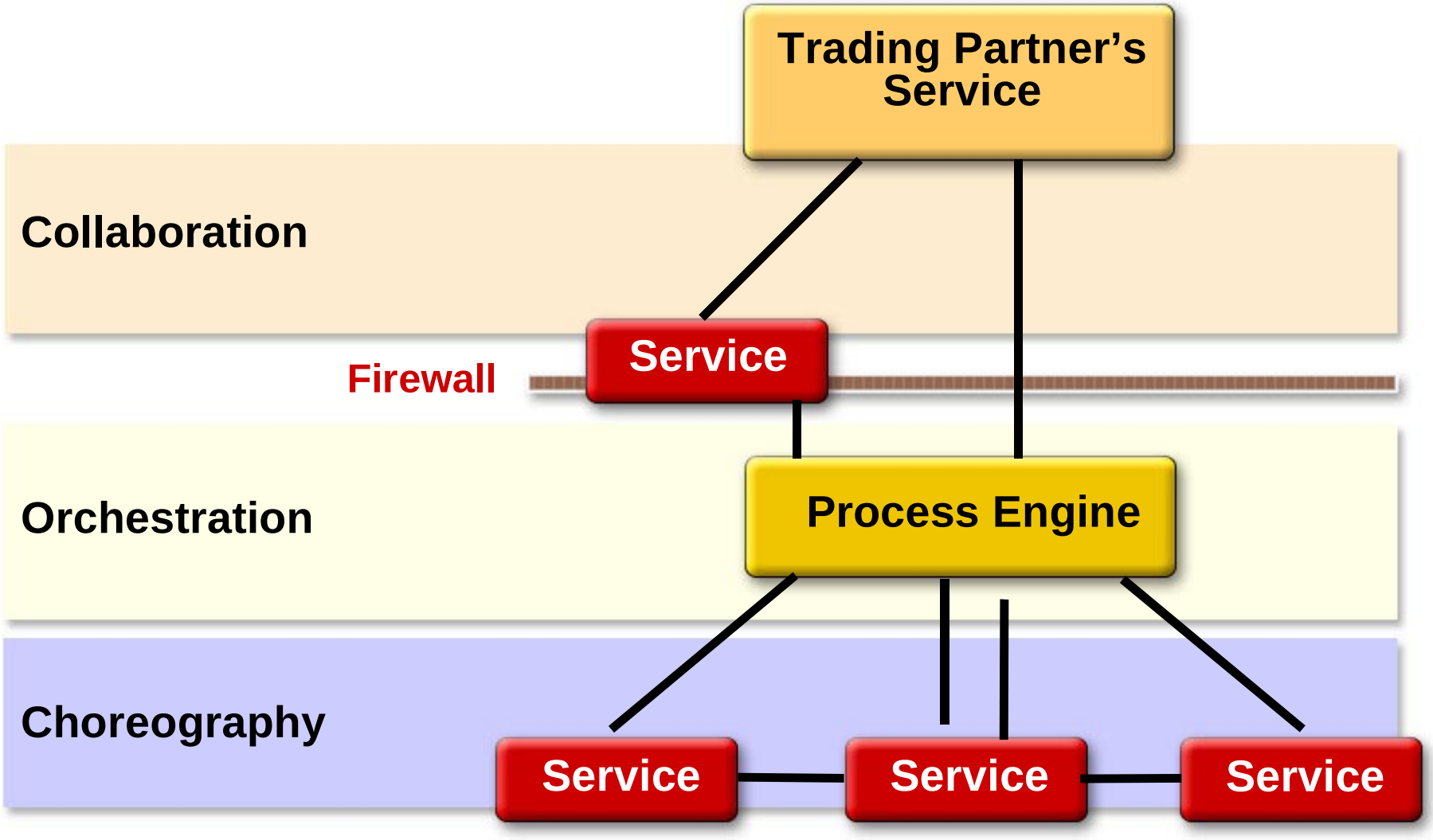

```
@Protocol (httpSoap=true, soapStyle=documentLiteral)
@TargetNamespace (namespace=http://schemas.myDomain.com/ws/)

public class MyWebService{
  @Operation
  public double zipDistance (String fromZip, String toZip){
    . . .
    return distance.getDistance(fromZip, toZip);
  }
  . . .
}
```

Orchestration

Definitions

- Collaboration
  - Interaction between two or more B2B partners
  - ebXML BPSS
- Orchestration
  - Running business processes
  - BPML, BPEL4WS
- Choreography
  - Observable behavior at service interface
  - WSDL, WSCI

Definitions (cont.)



Transaction

Transaction

- Web services have different characteristics
  - Long running business process
  - Multi enterprise and distributed (no single Transaction manager is present)
- ACID properties need to be loosened up for Web services
  - Traditional locking cannot be used for long running process
- BTP (Business Transaction Protocol) from OASIS

Reliable Messaging

WS-Reliability

- Co-authored by Sun, Hitach, Fujitsu, NEC, Oracle, Sonic
- OASIS TC formed
- Well-aligned with ebXML Messaging Service (MS)
  - ebXML MS provides reliable messaging for B2B
  - WS-Reliability is for light-weight reliable messaging within intranet

Security

XML & Web Services Security Schemes

Sun™ Tech Days

Sun microsystems

- XML Digital Signature
- XML Encryption
- XKMS (XML Key Management Specification)
- XACML (eXtensible Access Control Markup Language)
- SAML (Secure Assertion Markup Language)
- ebXML Message Service Security
- WS-Security
- Identity Management & Liberty Project

Resources

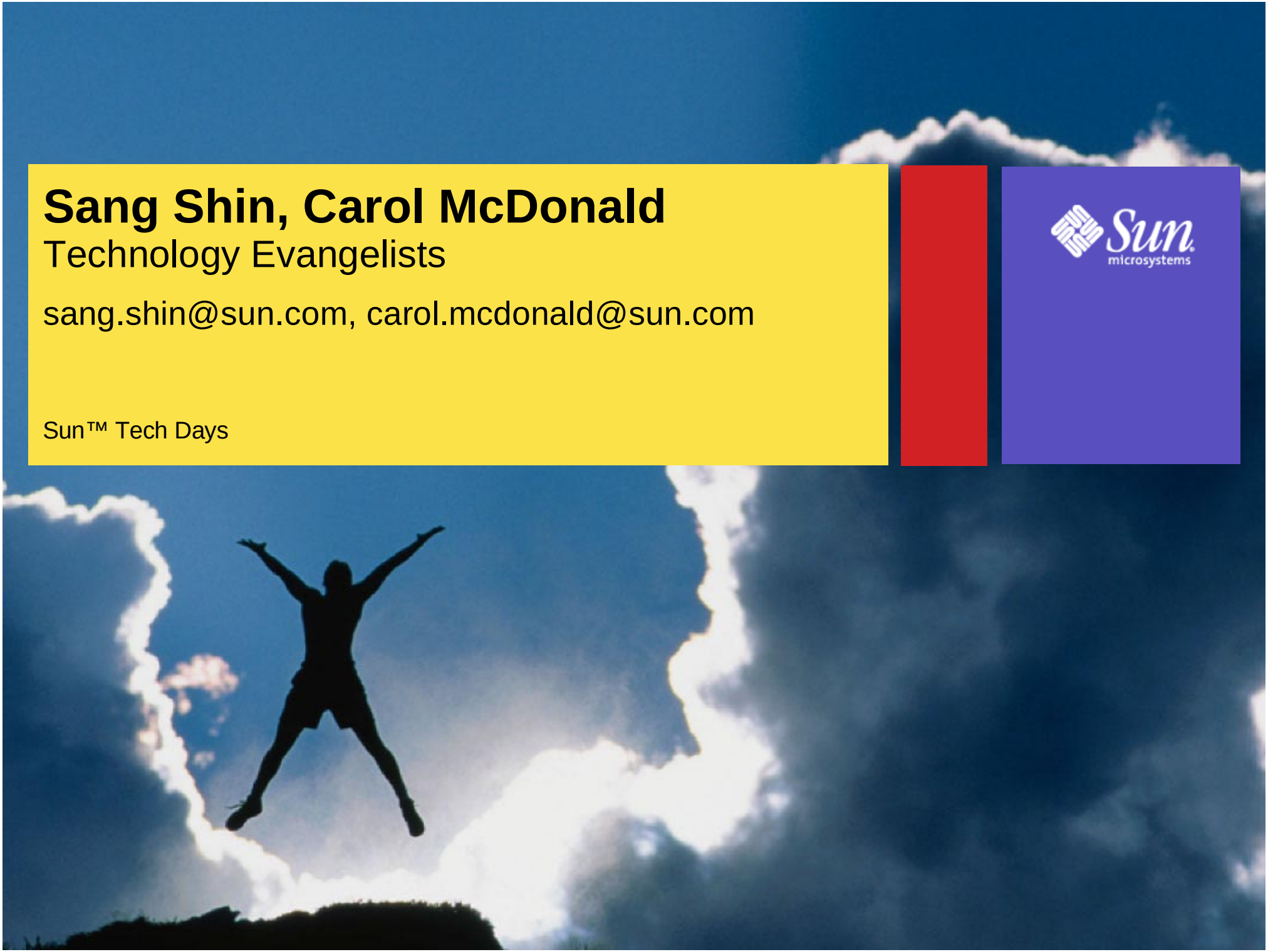
Resources

- Web Services Codecamp
  - developer.sun.com/dev/edu/camps/demos/webserviceslab/download.html
- Java Web Services Developer Pack Download
  - java.sun.com/webservices/downloads/webservicespack.html
- Java Web Services Developer Pack Tutorial
  - java.sun.com/webservices/downloads/webservicestutorial.html
- Sun ONE Application Server
  - wwws.sun.com/software/products/appsrvr/home_appsrvr.html
- Sun ONE Studio 5
  - wwws.sun.com/software/sundev/jde/buy/index.html

Resources

- J2EE Tutorial for Sun ONE Platform
  - java.sun.com/j2ee/1.3/docs/tutorial/doc/index.html
- Developing Amazon.com Web service client
  - developer.java.sun.com/developer/technicalArticles/WebServices/amazonws/
- Building Web services with Sun ONE Application Server
  - sunonedev.sun.com/building/tech_articles/jaxrpc/
- Sun ONE Studio Web services tutorial
  - wwws.sun.com/software/sundev/jde/examples/index.html
- JAX-RPC on the Sun ONE Web Services Platform Developer Edition
  - sunonedev.sun.com/building/tech_articles/jaxrpcs1.html

Q&A



Sang Shin, Carol McDonald

Technology Evangelists

sang.shin@sun.com, carol.mcdonald@sun.com

Sun™ Tech Days