Metro: JAX-WS, WSIT and REST

Carol McDonald
carol.mcdonald@sun.com

Agenda

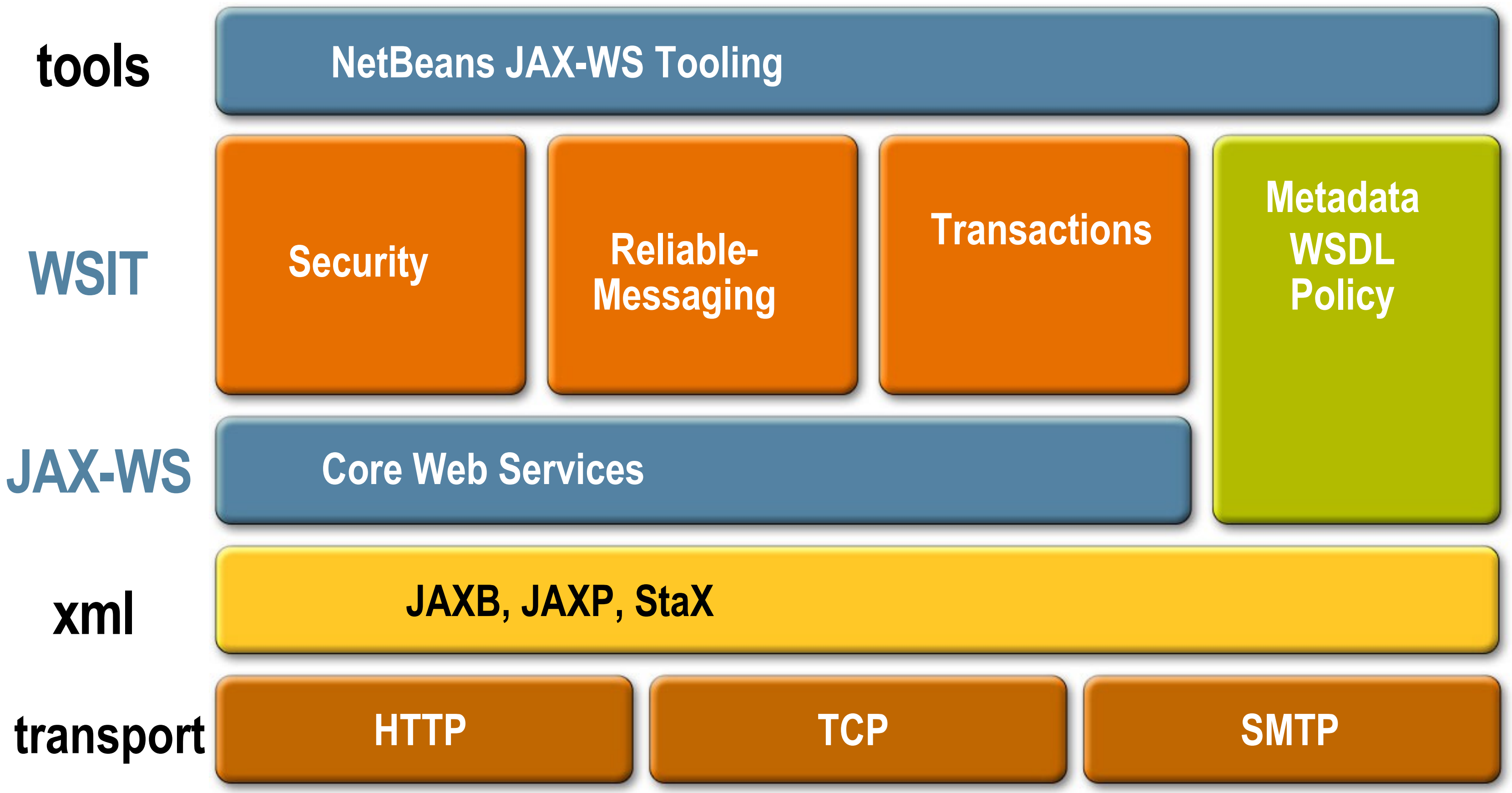
- Metro
  - JAX-WS
  - WSIT
- REST:
  - JAX-RS

Project Metro

- Project Metro = Java.net project for all of the components that make up the GlassFish WS stack
  - http://metro.dev.java.net
  - Can also be used outside of Glassfish
- key components of Metro:
  - JAX-WS and WSIT

Sun's Web Services Stack

Metro: JAX-WS , WSIT



JAXB = Java Architecture for XML Binding | JAX-WS = Java APIs for XML Web Services

Metro

- In addition to Glassfish,
- Can be used as WS stack with JBoss, WebLogic Server 10, Apache Tomcat, Jetty, and Java SE, TmaxSoft JEUS 6, Spring

Agenda

- Metro
  - > JAX-WS
  - > WSIT
- REST

JAX-WS

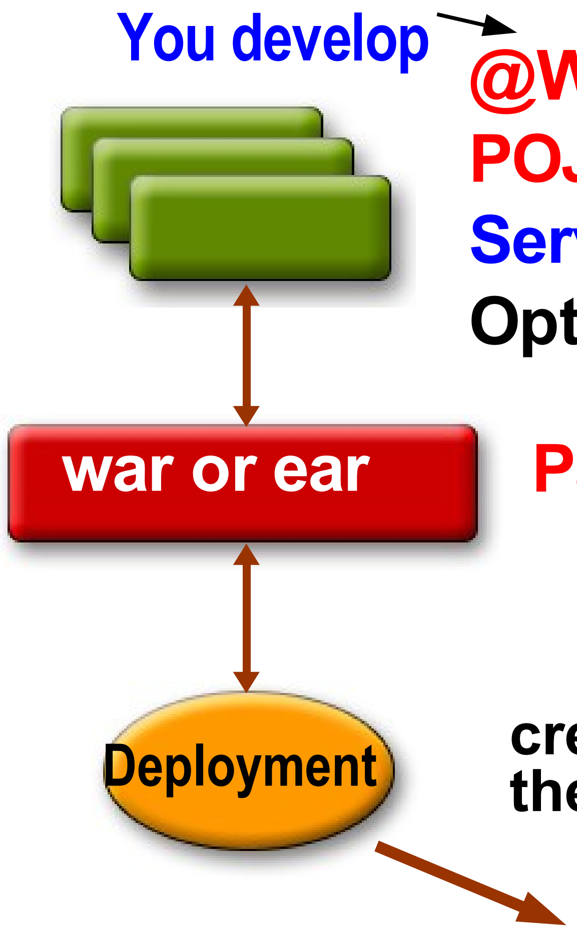
- easy to use Java API for XML Web Services
  - > Replaces JAX-RPC
- Descriptor-free programming
- Just Add annotation to plain old Java object (POJO)
- Layered Architecture
- SOAP 1.2
- Uses JAXB for data binding
- Part of Java SE 6 and Java EE 5 platforms

JAX-WS Standards

- JSR 224 Expert Group:
  - ATG BEA Capgemini Developmentor IBM Intalio IONA Motorola Nokia Novell NTT Oracle Pramati Red Hat SAP SeeBeyond Sonic Software Sun Tmax Trifork WebMethods

Developing a Web Service

Starting with a Java class



@WebService

POJO Implementation class

Servlet-based or Stateless Session EJB

Optional handler classes

Packaged application (war/ear file)

creates JAXB and JAX-WS files needed for the service

Service contract

Example: Servlet-Based Endpoint

```
@WebService
public class CalculatorWS {
    public int add(int a, int b) {
        return a+b;
    }
}
```

- All public **methods** become web service **operations**
- **WSDL/Schema generated at deploy time automatically**
- **Default** values for WSDL **service** name, etc.

Service Description default mapping

Java mapping -> WSDL:

```
<portType name="CalculatorWS">
    <operation name="add">
       <input message="tns:add"/>
       <output message="tns:addResponse"/>
    </operation>
</portType>
```

PORT TYPE = ABSTRACT INTERFACE

OPERATION = METHOD

MESSAGE = PARAMETERS AND RETURN VALUES

```
public class CalculatorWS{
     public int add(int i, int j){
    }
}
```

Customizability via Annotations

```
@WebService(
    name="Calculator",
    portName="CalculatorPort",
    serviceName="CalculatorService",
    targetNamespace="http://calculator.org"
)
public class CalculatorWS {
    @WebMethod(operationName=”addCalc”)
    public int add(@WebParam(name=”param1”) int a, int b) {
        return a+b;
    }
}
```

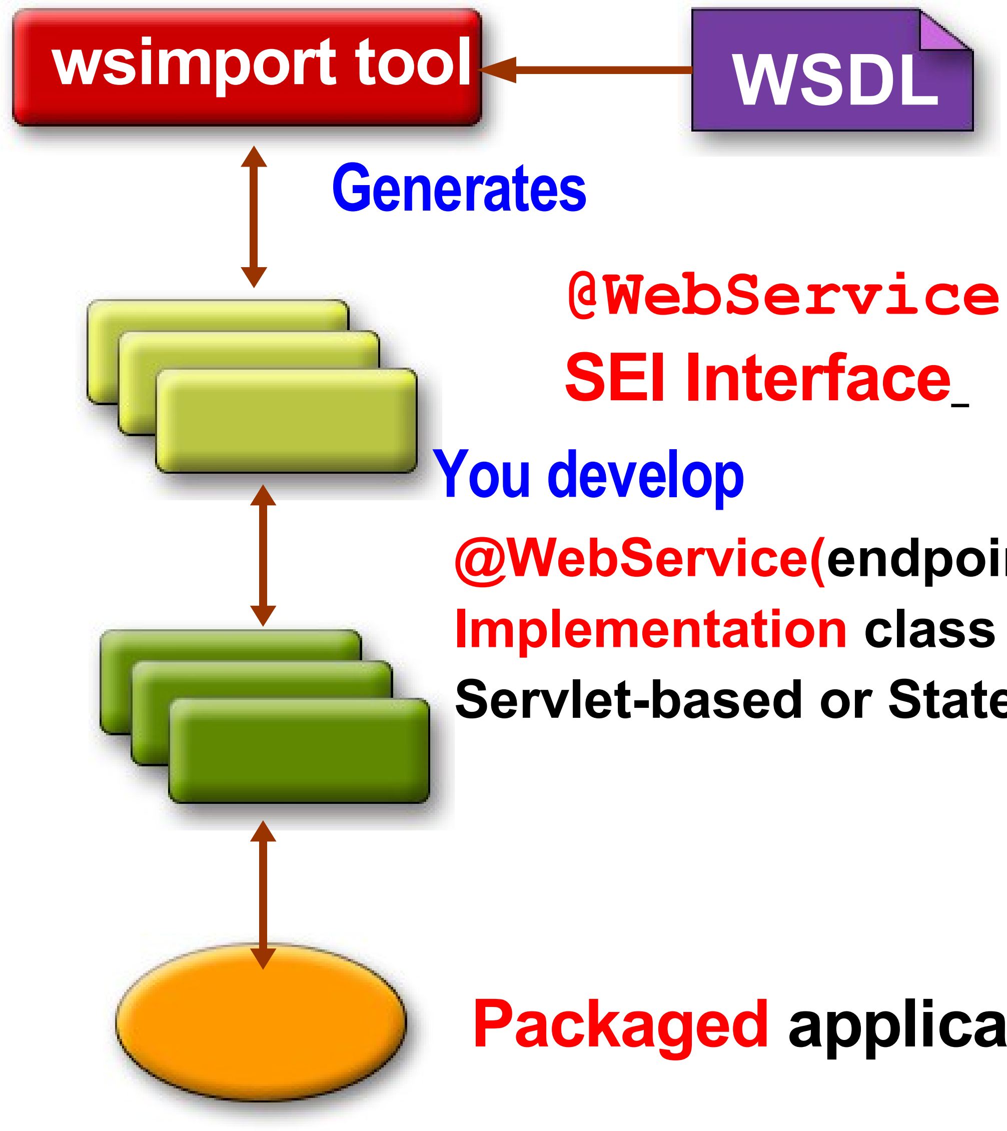
Example: EJB 3.0-Based Endpoint

```
@WebService
@Stateless
public class Calculator {

   public int add(int a, int b) {
      return a+b;
   }
}
```

- It's a regular EJB 3.0 component so it can use EJB features
  - **Transactions, security, interceptors**...

Developing a Web Service

Starting with a WSDL



Service contract

@WebService

SEI Interface

@WebService(endpointInterface="generated Service")

Implementation class

Servlet-based or Stateless Session EJB endpoint model

Packaged application (war/ear file)

Generating an Interface from WSDL

WSDL->Java generation:

```
<portType name="BankService">
   <operation name="getBalance">
      <input message="tns:getBalanceInput" />
      <output message="tns:getBalanceOutput" />
      <fault name="AccountException"
            message="tns:AccountException"/>
   </operation>
</portType>
```

PORT TYPE = INTERFACE
OPERATION = METHOD
MESSAGE = PARAMETERS

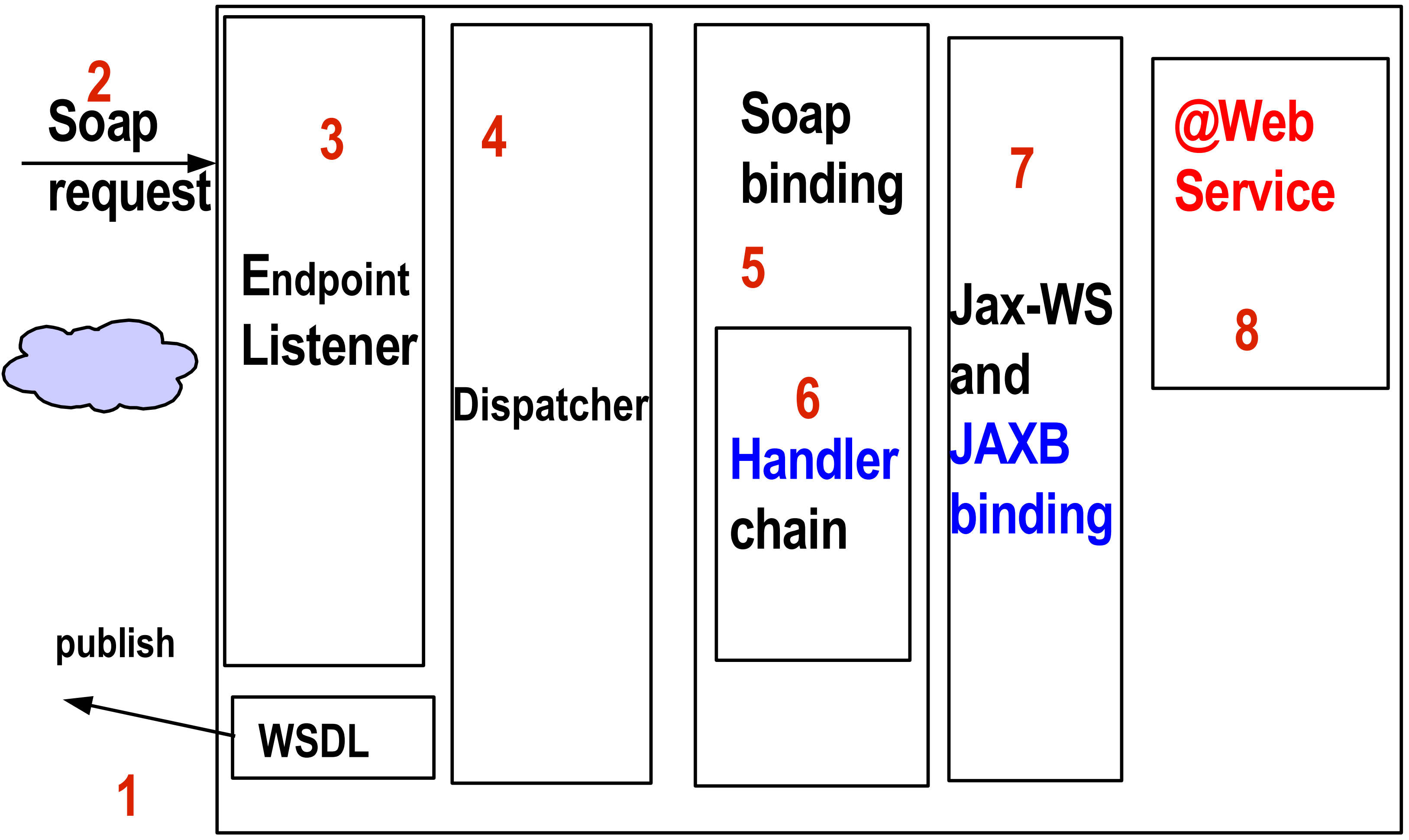
```
@WebService
public interface BankService{
  @WebMethod
  public float getBalance(String acctID,String acctName)
     throws AccountException;
}
```

Implementing a Web Service for a Generated Interface

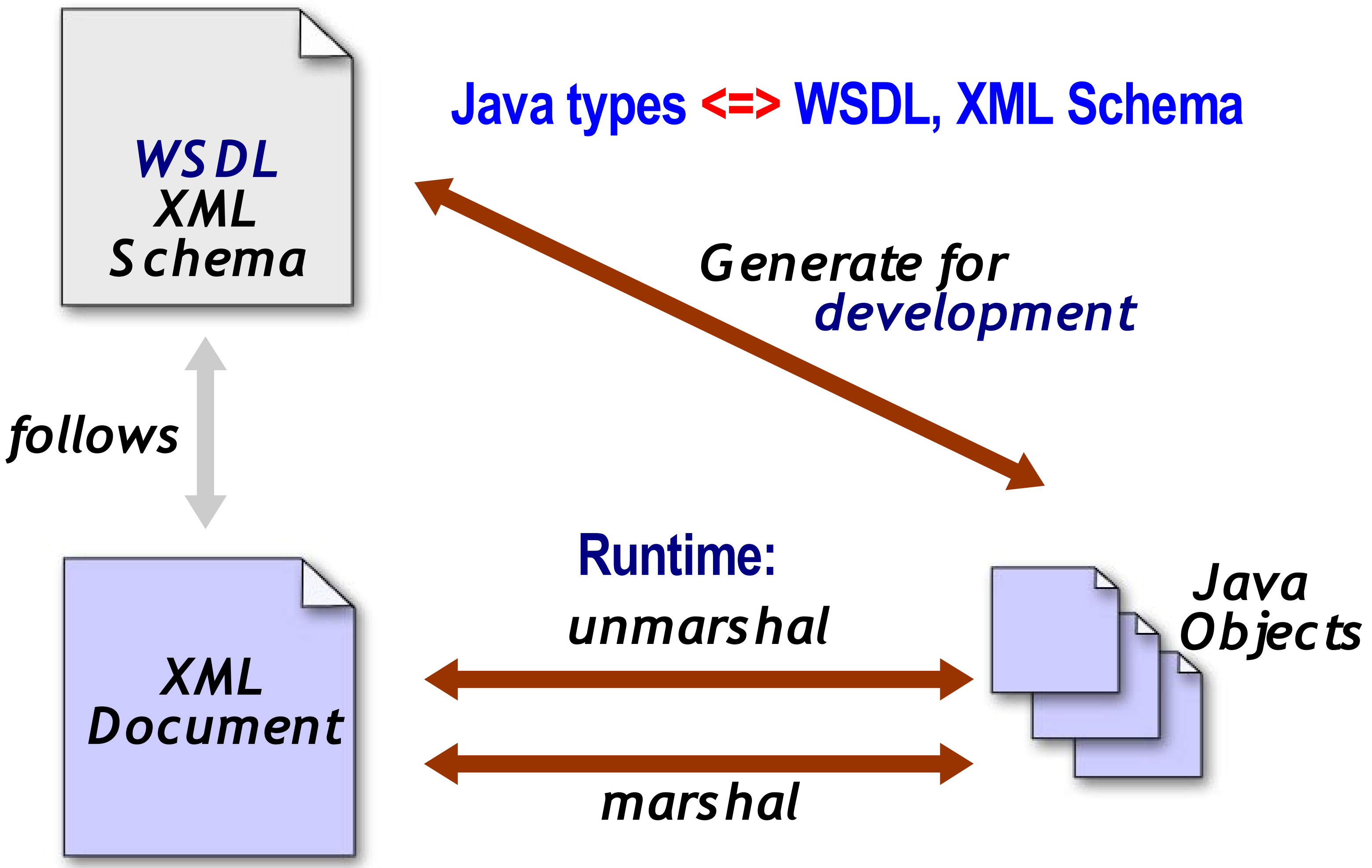
```
@WebService( endpointInterface="generated.BankService",
     serviceName="BankService")
public class BankServiceImpl implements BankService{
   ...
   public float getBalance(String acctID, String acctName)
     throws  AccountException {

        // code to get the account balance

       return theAccount.getBalance();
   }
}
```

Server Side

CalculatorWS Web Service



JAX-WS uses JAXB for data binding



Runtime: JAXB un/marshalls the payload

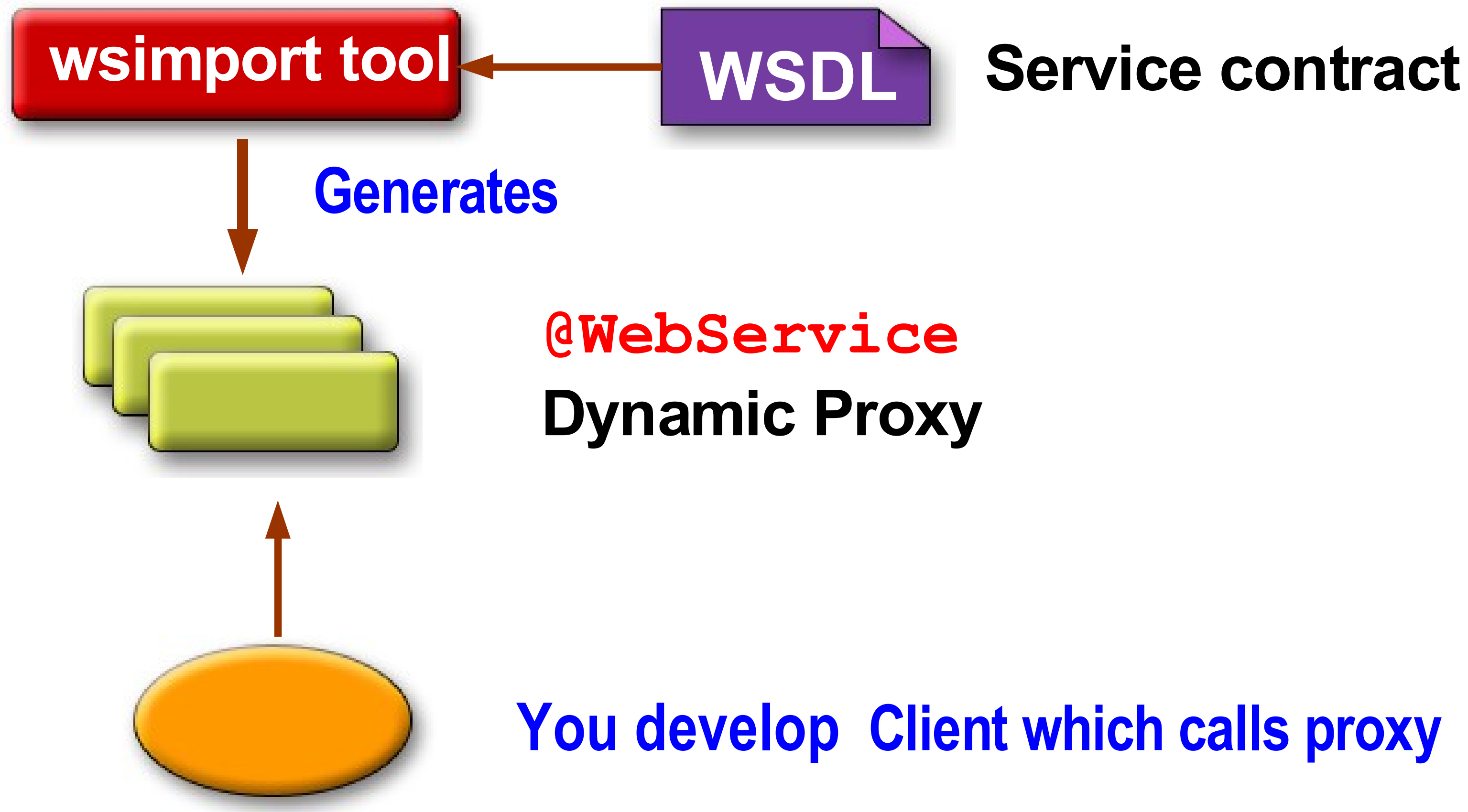
JAXB XML schema to Java mapping

```java
package calculator.jaxws;
import javax.jws.WebService;

@XmlRootElement(name = "add")
public class add{

 @XmlElement(name = "i")
 private int i;

 @XmlElement(name = "j")
 private int j;

 public int getI() {
   return this.i;
 }
 public void setI(int i){
        this.i = i;
 }
}
```

```xml
<?xml version="1.0" encoding="UTF-8"
    standalone="yes"?>
<xs:schema version="1.0" >
 <xs:element name="add"
   type="tns:add"/>

 <xs:element name="addResponse"
   type="tns:addResponse"/>

 <xs:complexType name="add">
    <xs:sequence>
      <xs:element name="i"
                  type="xs:int"/>
      <xs:element name="j"
                  type="xs:int"/>
    </xs:sequence>
 </xs:complexType>

</xs:schema>
```

demo

Client-Side Programming



Example: Java SE-Based Client

Factory Class

Business Interface

Get Proxy Class

```
CalculatorService svc = new CalculatorService();

Calculator proxy = svc.getCalculatorPort();
int answer = proxy.add(35, 7);
```

- code is fully portable
  - CalculatorService is defined by the specification
  - Internally it uses a delegation model

WSDL Java mapping

```
package calculator;

import javax.jws.WebService;

@WebService
public class CalculatorWS{

 public int add(int i, int j)
   {
     return i + j;
   }

}
```

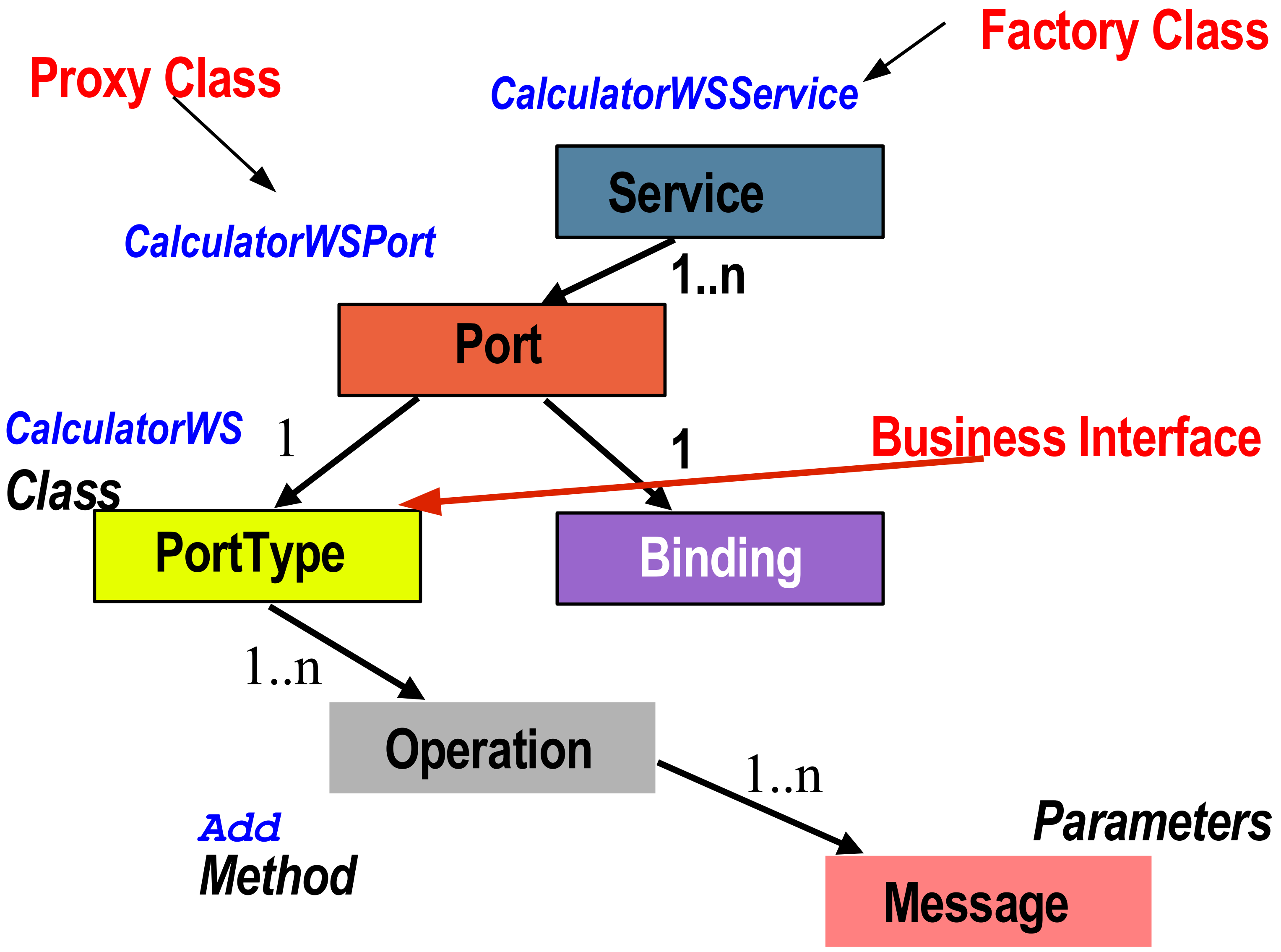
```
<portType name="CalculatorWS">
  <operation name="add">
    <input message="tns:add"/>
    <output message="tns:addResponse"/>
  </operation>
</portType>

<binding name="CalculatorWSPortBinding"
    type="tns:CalculatorWS">
  <soap:binding transport="soap/http"
    style="document"/>
  <operation name="add">
    ....
  </operation>
</binding>

<service name="CalculatorWSService">
 <port name="CalculatorWSPort"
    binding="tns:CalculatorWSPortBinding">
 <soap:address location=
      "http://CalculatorWSService" />
  </port>
</service>
```

Business Interface

Factory Class

Proxy Class

WSDL to Dynamic Proxy mapping



Example: Java EE Servlet Client

No Java Naming and Directory Interface™ API !

```
public class ClientServlet extends HttpServlet {

 @WebServiceRef(wsdlLocation = "http://.../CalculatorWSService?wsdl")
 private CalculatorWSService service;

   protected void processRequest(HttpServletRequest req,
                                 HttpServletResponse resp){
       CalculatorWS proxy = service.getCalculatorWSPort();
       int i = 3; j = 4;
       int result = proxy.add(i, j);
       . . .
   }
}
```

demo

SOAP Request

http://localhost:8080/CalculatorWSApplication/CalculatorWSService

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope
   xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
     <S:Header/>
     <S:Body>
         <ns2:add xmlns:ns2="http://calculator.me.org/">
             <i>4</i>
             <j>3</j>
         </ns2:add>
     </S:Body>
</S:Envelope>
```

SOAP Response

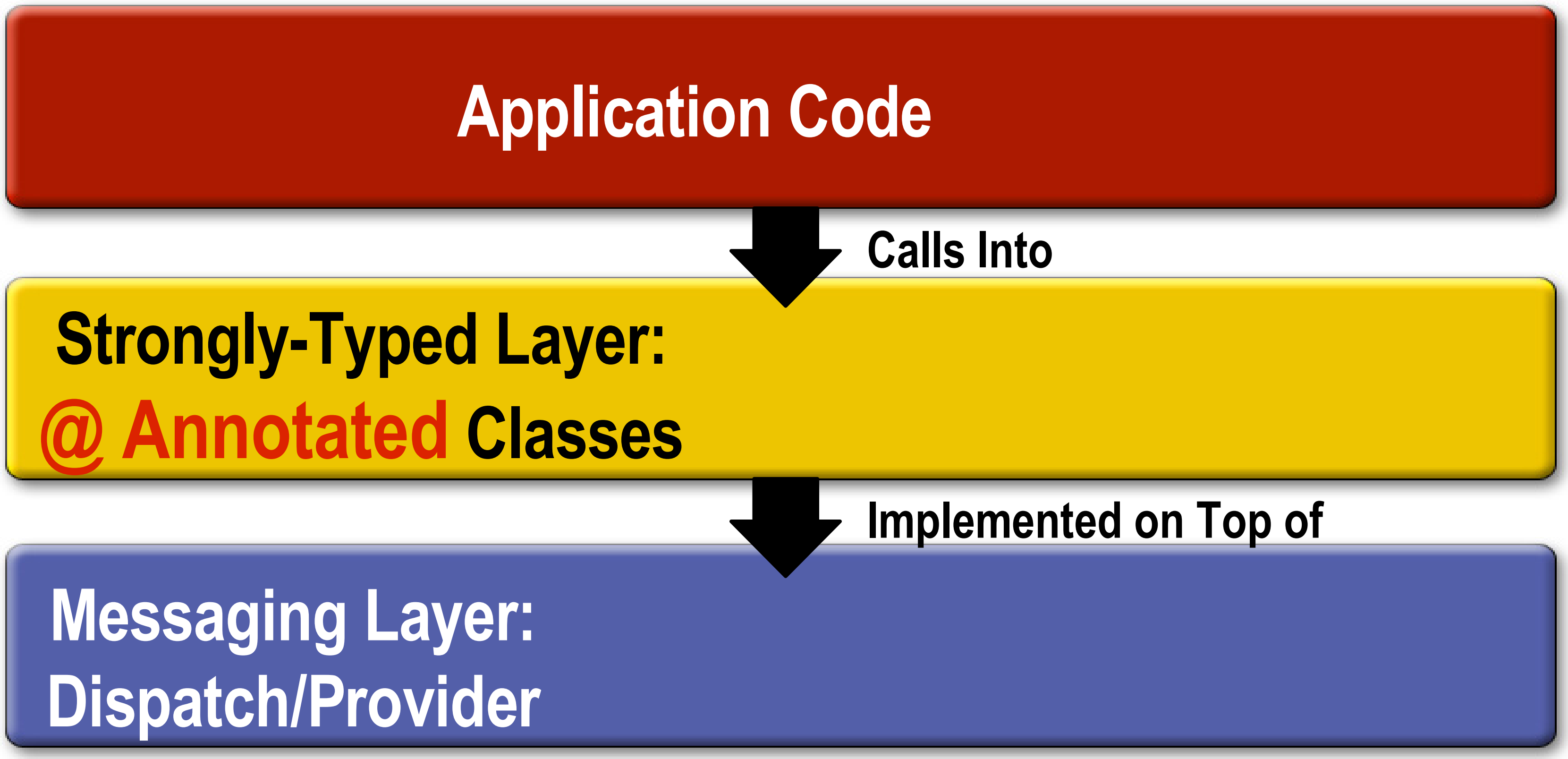
```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope
  xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Body>
        <ns2:addResponse xmlns:ns2="http://calculator.me.org/">
            <return>7</return>
        </ns2:addResponse>
    </S:Body>
</S:Envelope>
```

JAX-WS Layered Architecture

Upper layer Easy to use with annotations

Lower layer, API-based, more control

  For advanced scenarios



Lower Level

- Lower layer, API-based, more control:
- Client XML API: Dispatch Interface
  - > one-way and asynch calls available
- Server XML API: Provider Interface:
  - > Can use JAXB, JAXP, SAAJ to get message contents
- Message or Payload access
- May be used to create RESTful clients/services

Client-side Messaging API:

Dispatch Interface one-way and asynch calls available:

```
// T is the type of the message
public interface Dispatch<T> {

   // synchronous request-response
   T invoke(T msg);

   // async request-response
   Response<T> invokeAsync(T msg);
   Future<?> invokeAsync(T msg, AsyncHandler<T> h);

   // one-way
   void invokeOneWay(T msg);
}
```

Client-side Example:
Dispatch Using PAYLOAD

```
import javax.xml.transform.Source;
import javax.xml.ws.Dispatch;

private void invokeAddNumbers(int a,int b) {

   Dispatch<Source> sourceDispatch = service.createDispatch
       (portQName, Source.class, Service.Mode.PAYLOAD);

   StreamSource request =new StringReader(xmlString);

   Source result = sourceDispatch.invoke(request));

   String xmlResult = sourceToXMLString(result);

}
```

Server-side Messaging API: Provider

```
// T is the type of the message
public interface Provider<T> {

   T invoke(T msg, Map<String,Object> context);

}
```

- Message or Payload access
- Use @ServiceMode to select a mode for the message type

Server-sideExample:
Payload Mode, No JAXB

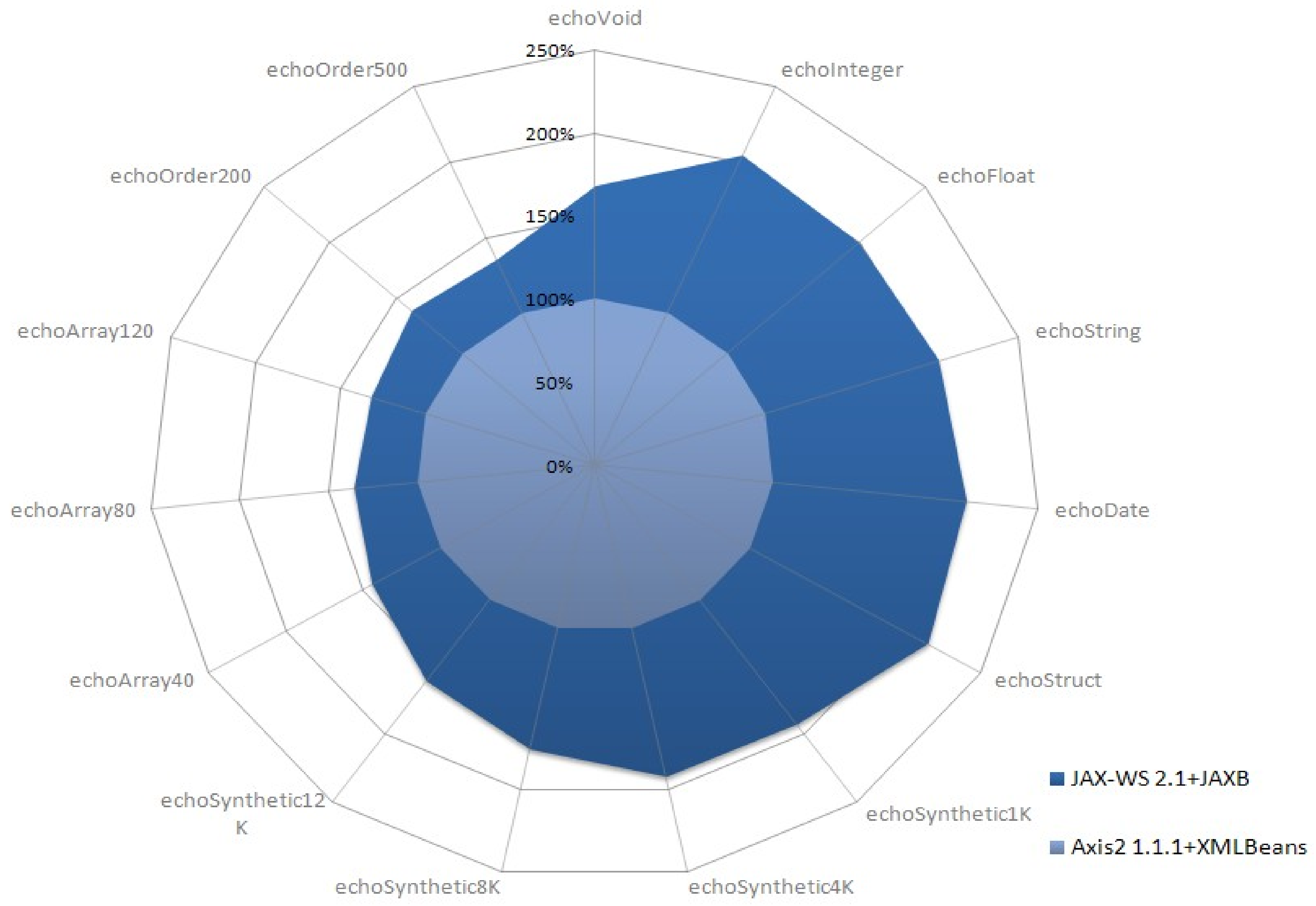
```
@ServiceMode(Service.Mode.PAYLOAD)
public class MyProvider implements Provider<Source> {

  public Source invoke(Source request,
                      Map<String,Object> context) {
   // process the request using XML APIs, e.g. DOM
   Source response = ...

   // return the response message payload
   return response;
  }
}
```

JAX-WS Commons
https://jax-ws-commons.dev.java.net/

Convenient Extensions, utility code , useful Plugins :

- Spring support
- Stateful Web Service
- Multiple Service Instances
  - HTTP Session-scope service
  - Thread scope service
- JSON Encoding
- Server-Side Asynchrony

JAX-WS 2.1 Performance vs Axis 2.1



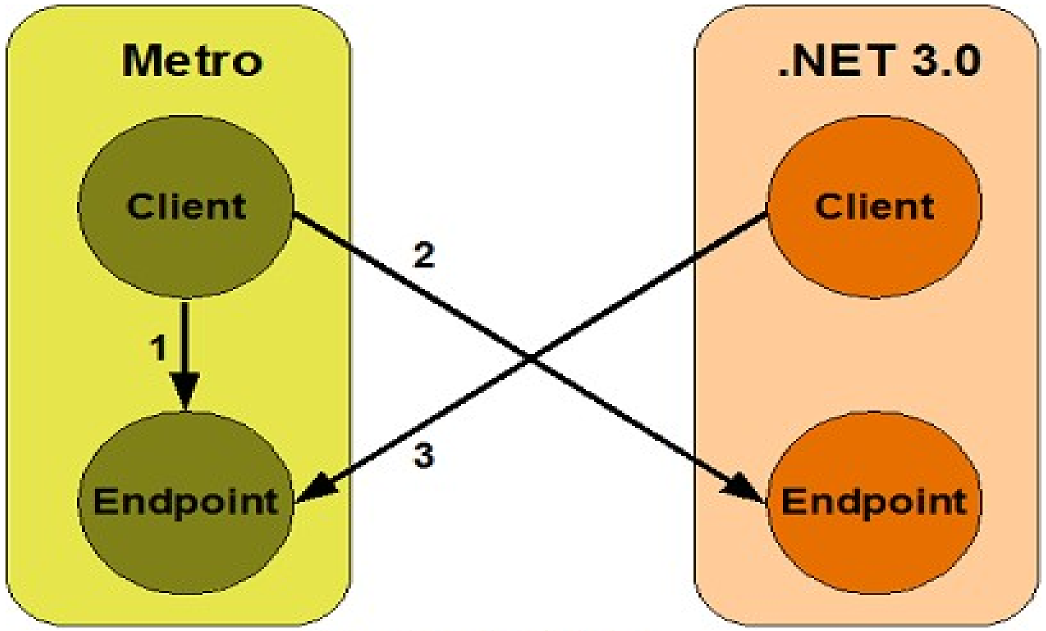
Agenda

- Metro
- JAX-WS Standards
- WSIT
- REST

WSIT:

Web Services Interoperability Technology

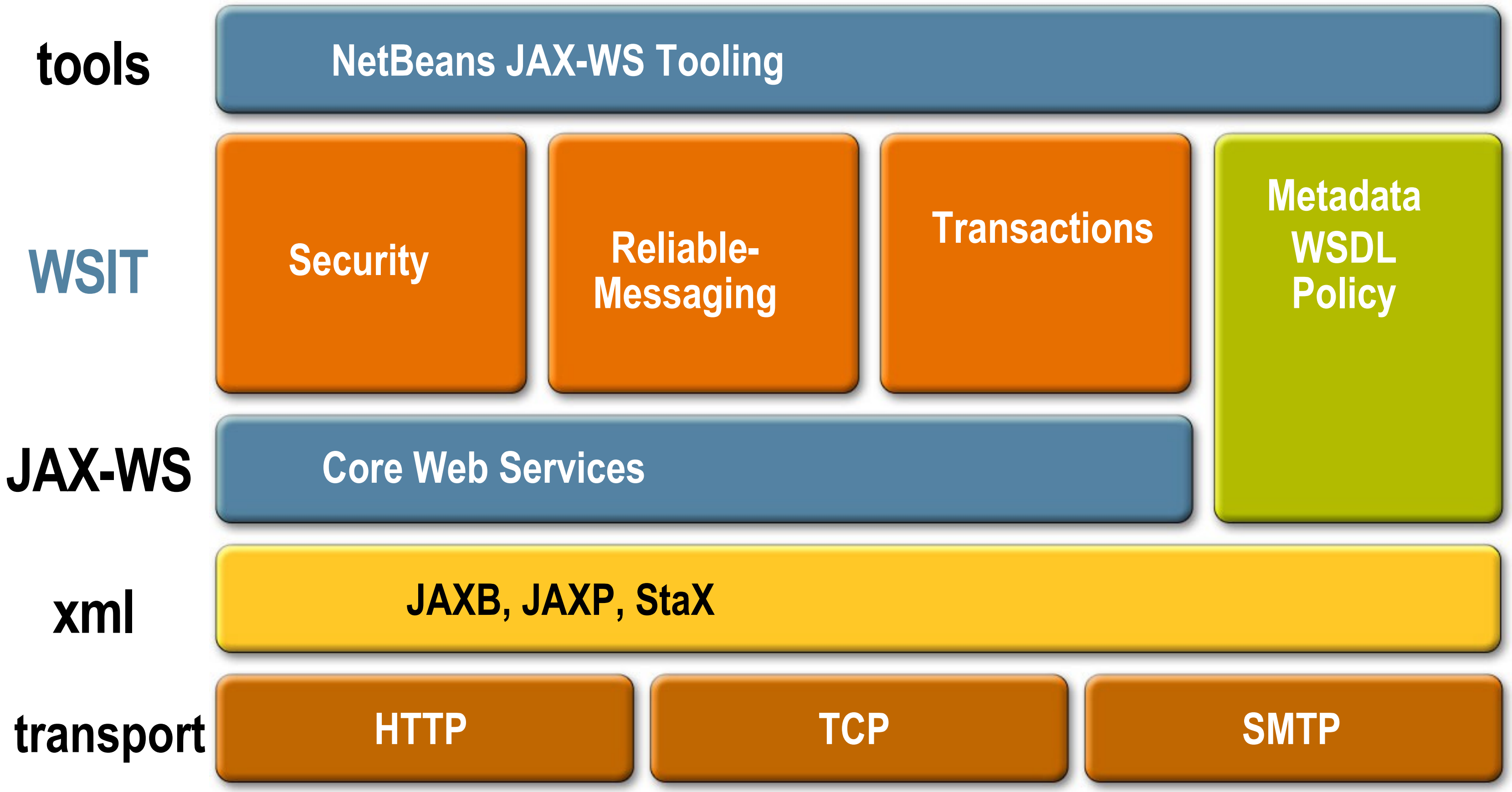
- Complete WS-* stack
- Enables interoperability with Microsoft .NET 3.0



Project Metro
metro.dev.java.net

Sun's Web Services Stack

Metro: JAX-WS , WSIT



JAXB = Java Architecture for XML Binding | JAX-WS = Java APIs for XML Web Services

Project Tango Features

WSIT (Web Services Interoperability Technology)

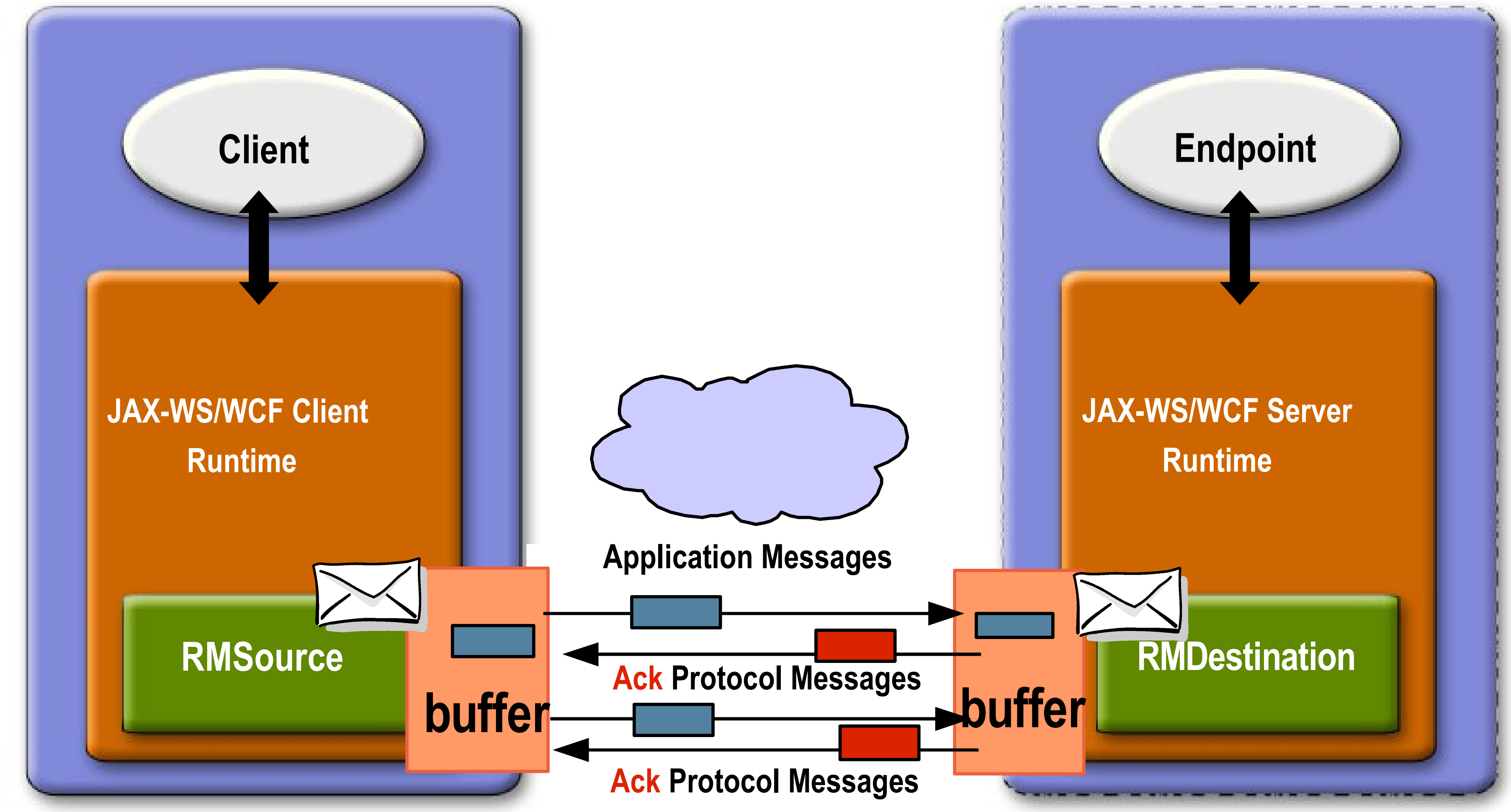
Enables interoperability with Microsoft .NET 3.0

- Bootstrapping communication
- End-to-end reliability
- Atomic transactions
- End-to-end security
- Trust
- Optimized security

Metro WSIT
Reliable Messaging

WS-ReliableMessaging

- **RMSource** handles sending and re-sending
- **RMDestination** handles reconstructing the stream of messages

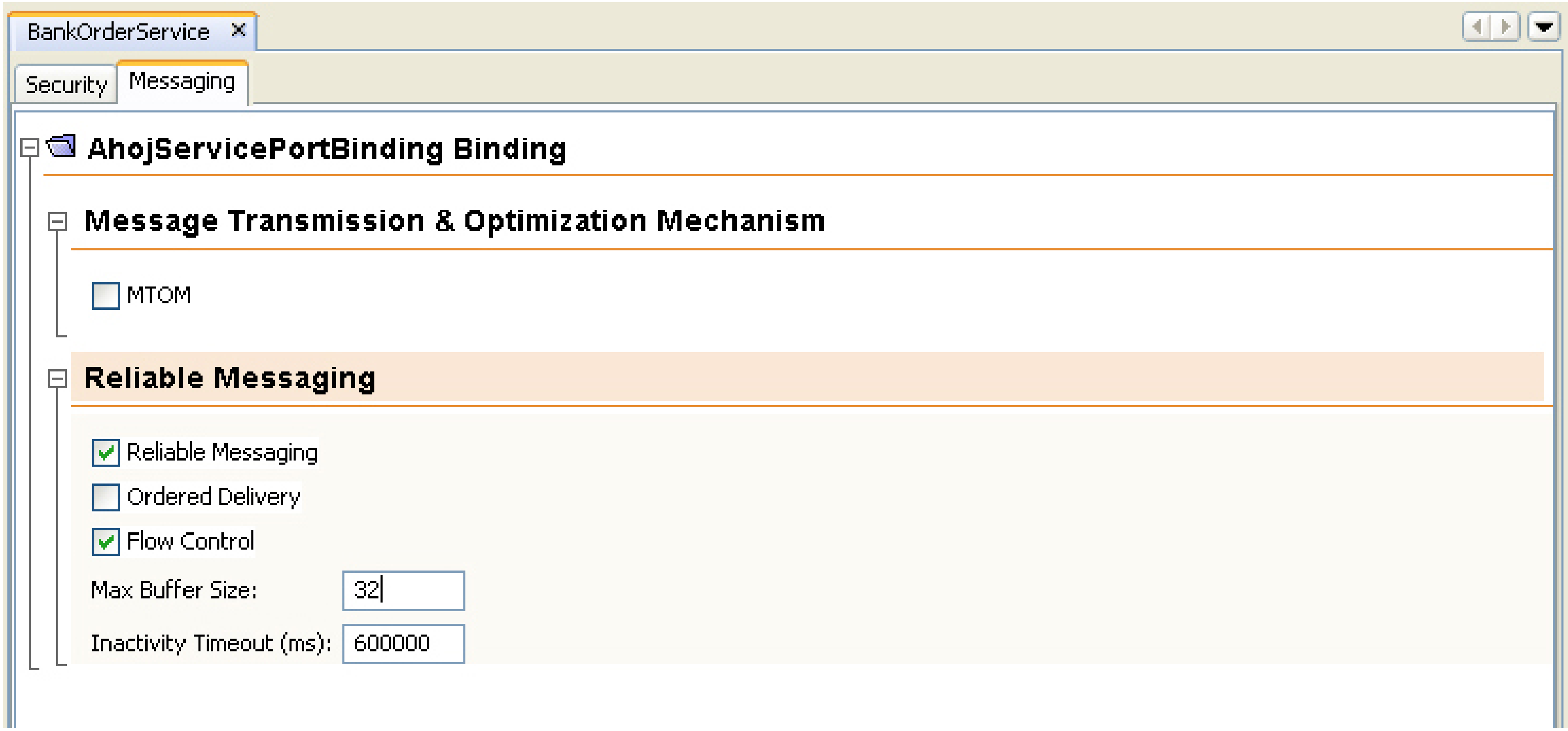


End-to-End Reliability

WS-ReliableMessaging

- Brings reliability to SOAP (protocol) layer
- Transparent to application
- Delivery assurance
  - At least once
  - At most once
  - In order

Configuration with NetBeans



BankOrderService ✕

Security | Messaging

AhojServicePortBinding Binding

Message Transmission & Optimization Mechanism

☐ MTOM

Reliable Messaging

☑ Reliable Messaging

☐ Ordered Delivery

☑ Flow Control

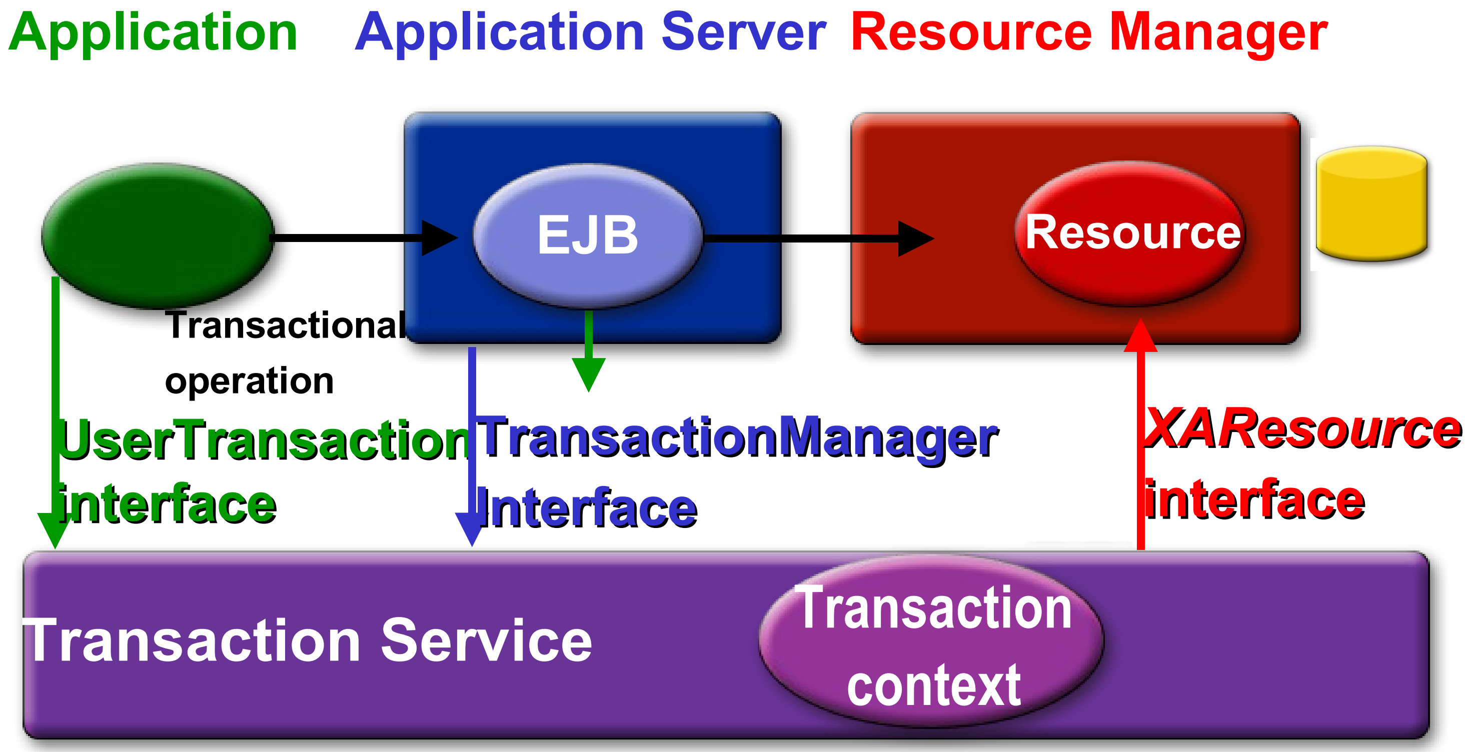
Max Buffer Size:	32
Inactivity Timeout (ms):	600000

Reliable Transport Alternatives

- SOAP messages are transport agnostic
  - Change the transport, change the binding
- Metro Transports (Htttp standard):
  - JMS
  - SMTP
  - SOAP over TCP
  - For more information
  - https://jax-ws.dev.java.net/transport.html

Metro WSIT Transactions

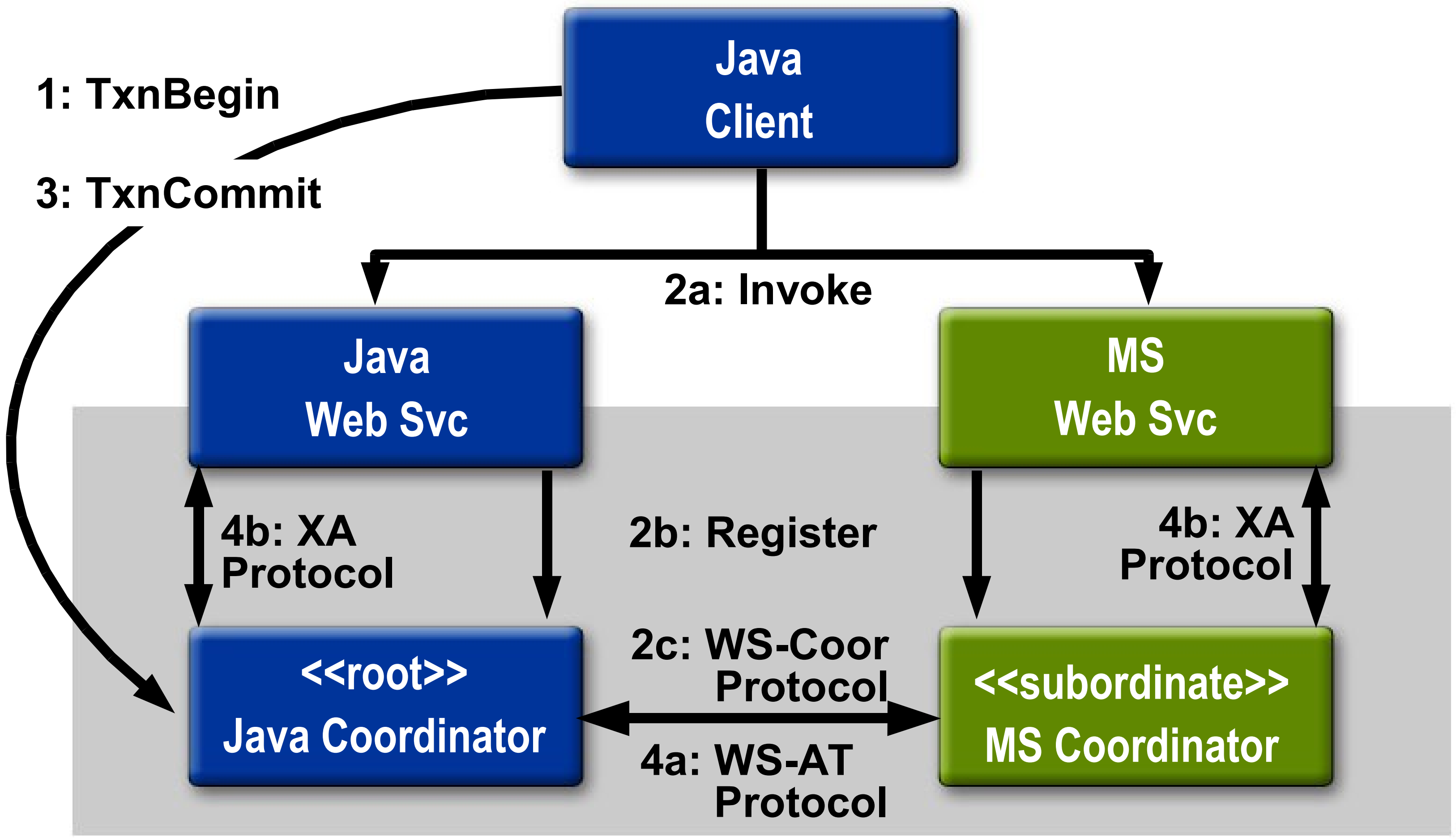
Java™ Transaction Service



WSIT Transaction coordination

- WS-AtomicTransaction defines set of transaction coordination protocols
  - All-or-nothing web service operations
  - Two-phase commit protocol
- WS-Coordination to coordinate the actions of distributed web services
  - For Commit: Coordinator asks if each system is ready to complete
    - If all concur, coordinator tells systems to complete
    - Otherwise, coordinator tells systems to rollback
  - Metro supports the *Durable two-phase Commit* (Durable 2PC) protocol

WSIT and WCF
Co-ordinated transaction



WSIT Support on Transaction

- an Atomic Transaction Context is created the first time a transacted Web service operation is invoked within a JTA transaction scope

```
@Resource
javax.transaction.UserTransaction ut;

ut.begin();
bankWebService.makeWithdrawl();
...
ut.commit();.
```

Transaction Context created

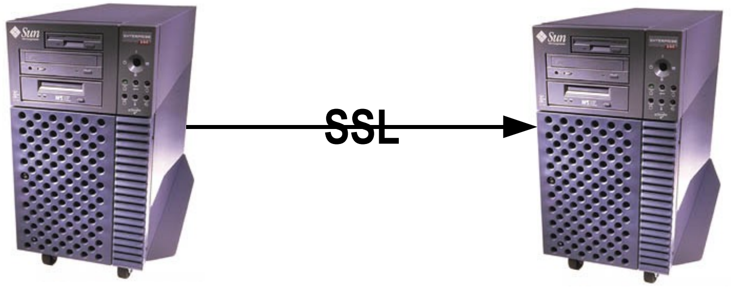
Transactions in Action

```
@WebService
@Stateless

public class Wirerer {

    @TransactionAttribute(REQUIRED)
    void wireFunds(...) throws ... {
        websrvc1.withdrawFromBankX(...);
        websrvc2.depositIntoBankY(...);
    }
}
```

Metro WSIT Security

Security

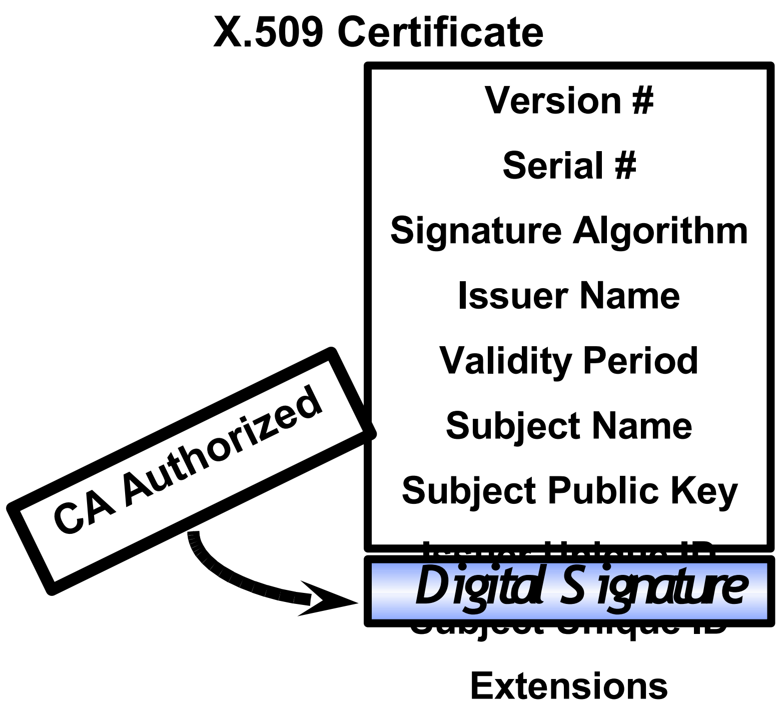
Before WS-Security



- SSL/HTTPS
- Security at transport layer
- Point-to-point
- Encrypts session

Digital Certificate

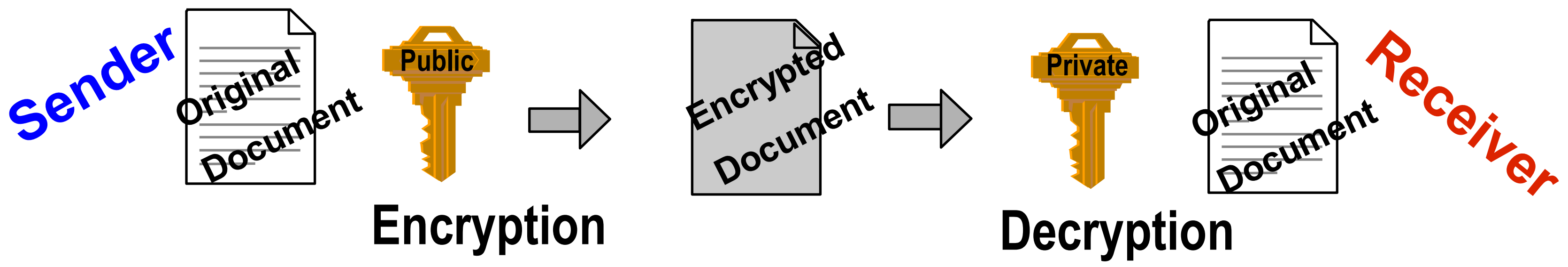
Identity data signed by a Certification Authority. Provides a Trusted source of identification.



Digital ID

- Electronic Proof of Identity
- Issued and signed by Certifying Authority
- Public, Private keys
- Makes security protocols work
  - SSL

Encryption

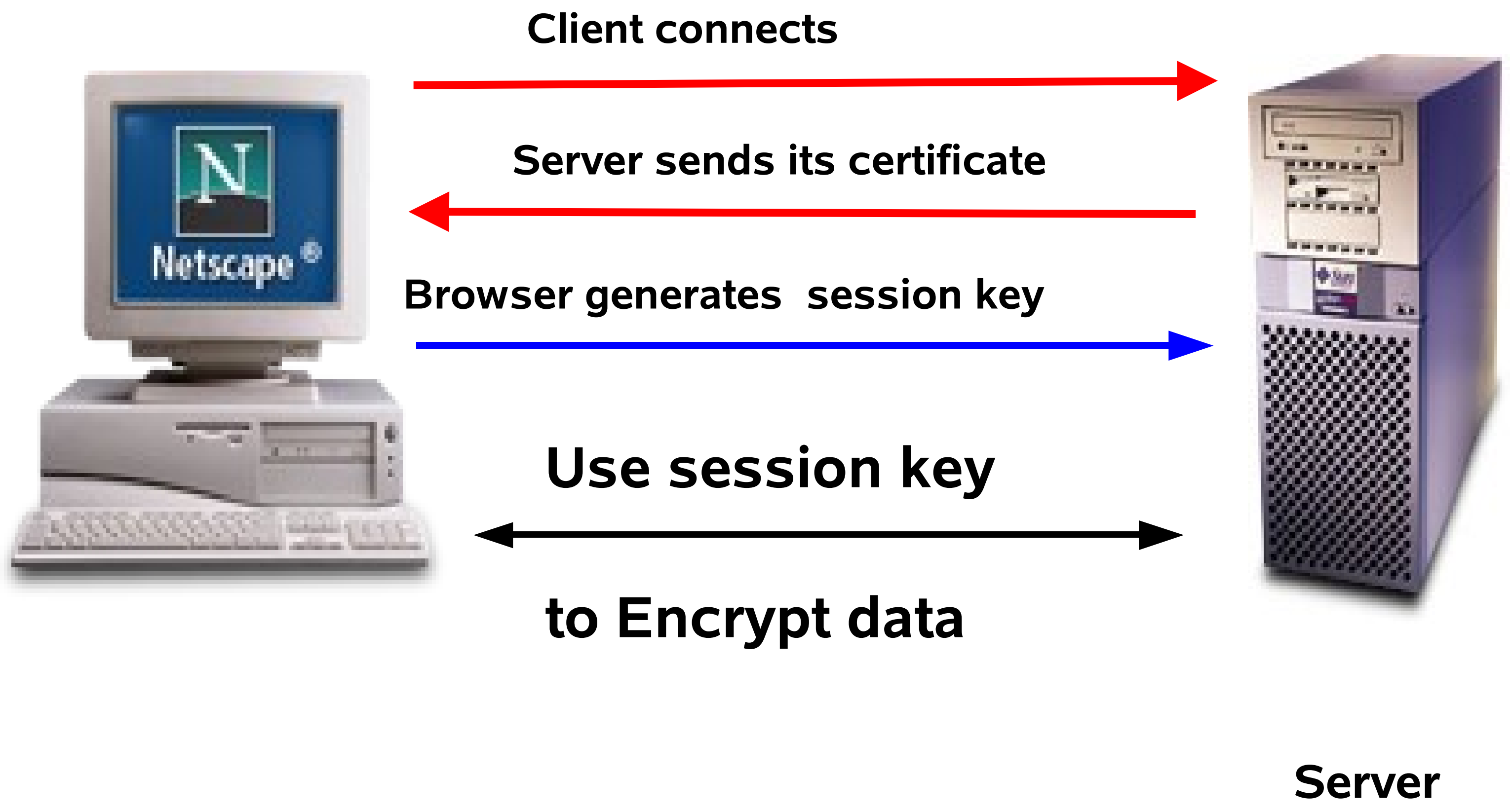


Receiver
Public
Key

Receiver
Private
Key

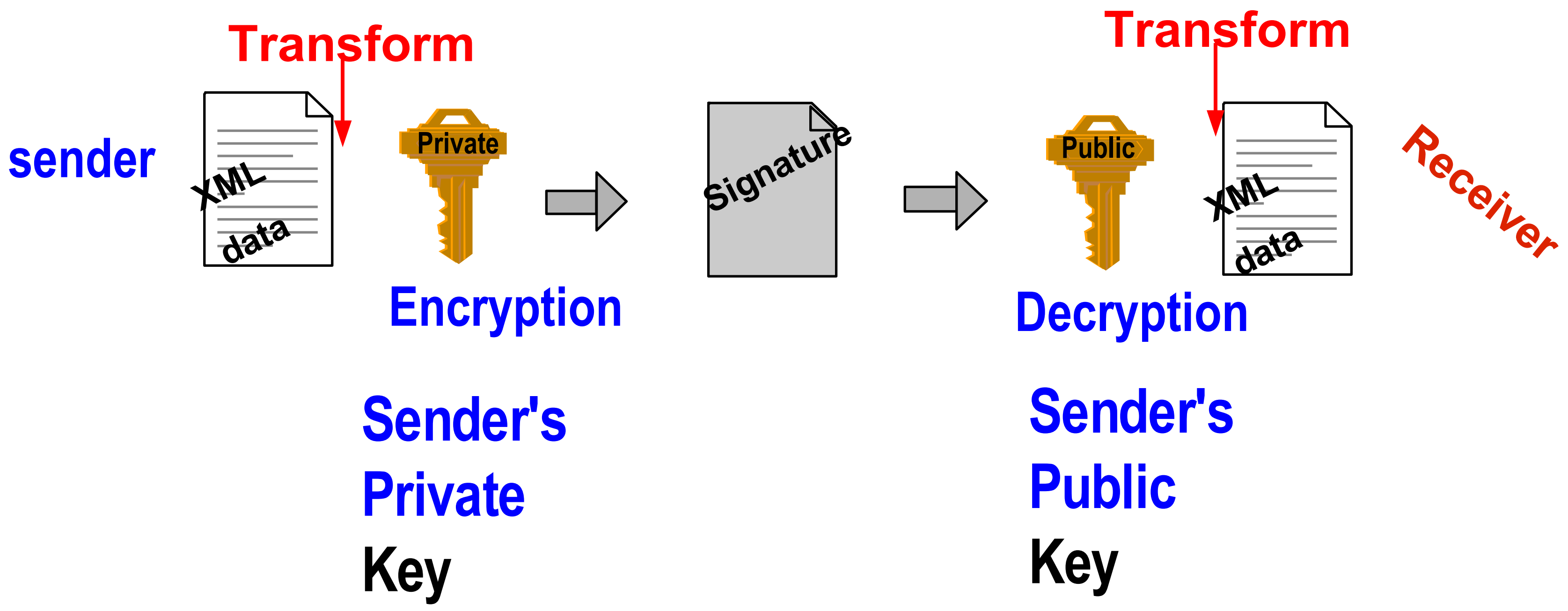
- XML Encryption (data confidentiality)
  - How digital content is encrypted

SSL Key Exchange



· Browser and Server use Session Key$_B$ to encrypt all data exchanged over the Internet

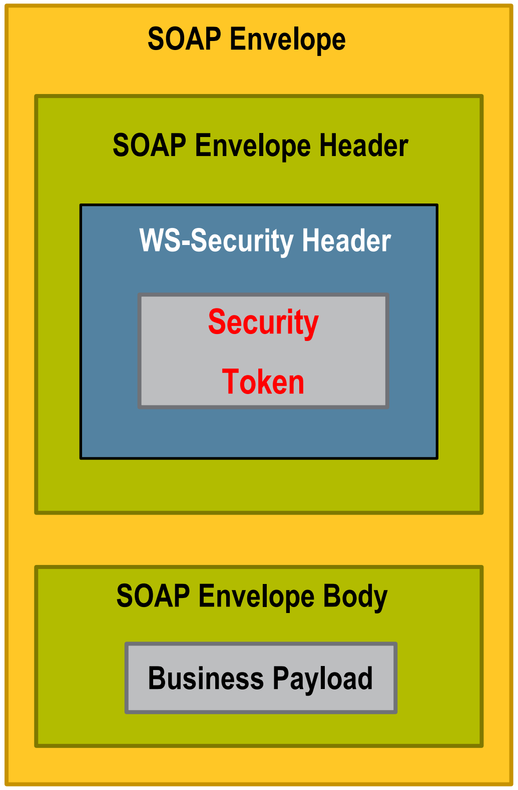
Digital Signature



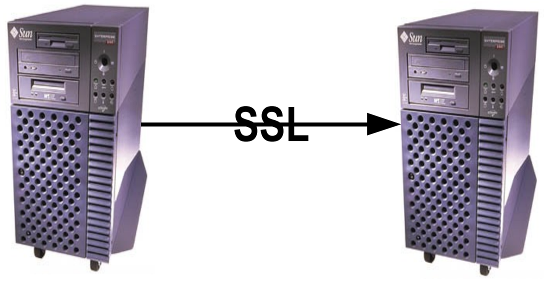
- XML Signature
- Bind the sender’s identity to an XML document

WS-Security: SOAP Message Security

- WS-Security defines:
  - Encrypting and signing message parts:
    - XML Signature and XML Encryption in SOAP Header
  - How to pass security tokens
  - (token=security-related information)
    - X.509 certificates
    - Kerberos tickets
    - UserName token

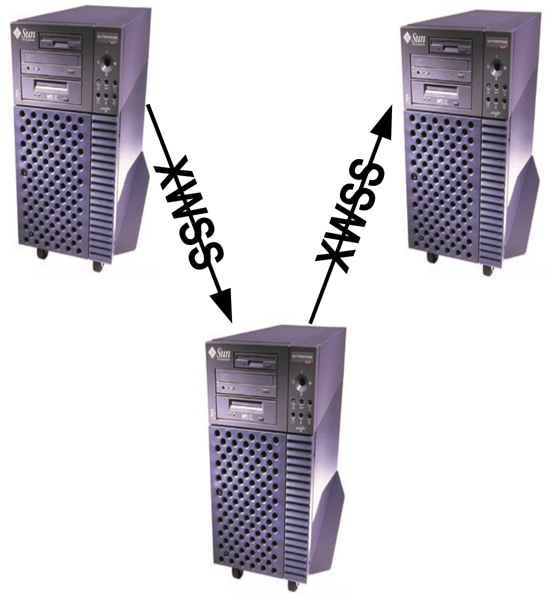


Security



Before WS-Security

- SSL/HTTPS
- Security at transport layer
- All or nothing granularity
- Point-to-point

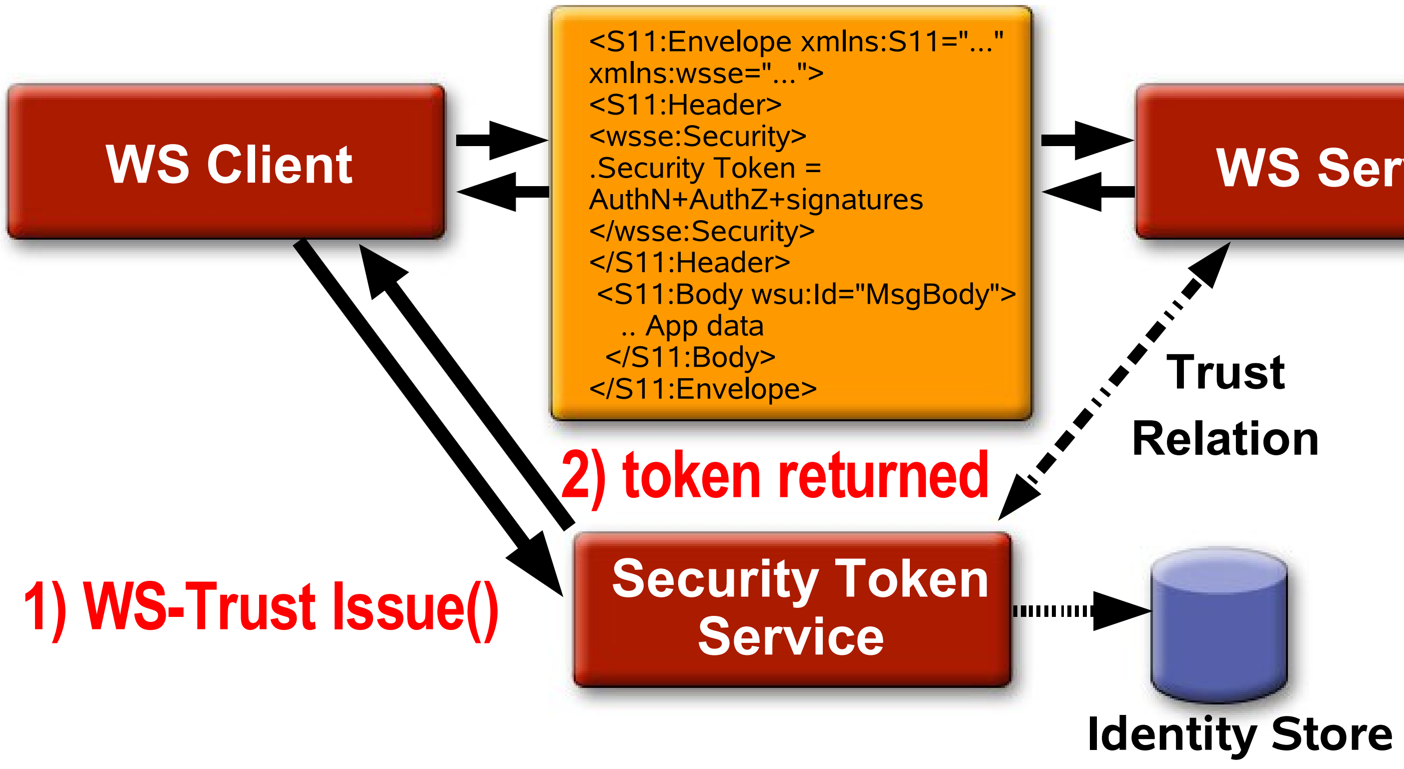


WS-Security

- Security at SOAP (protocol) layer
- Only sign/encrypt part of msg
- Works on non-TCP/IP transports
- Work with intermediaries
- XML Signature/Encryption

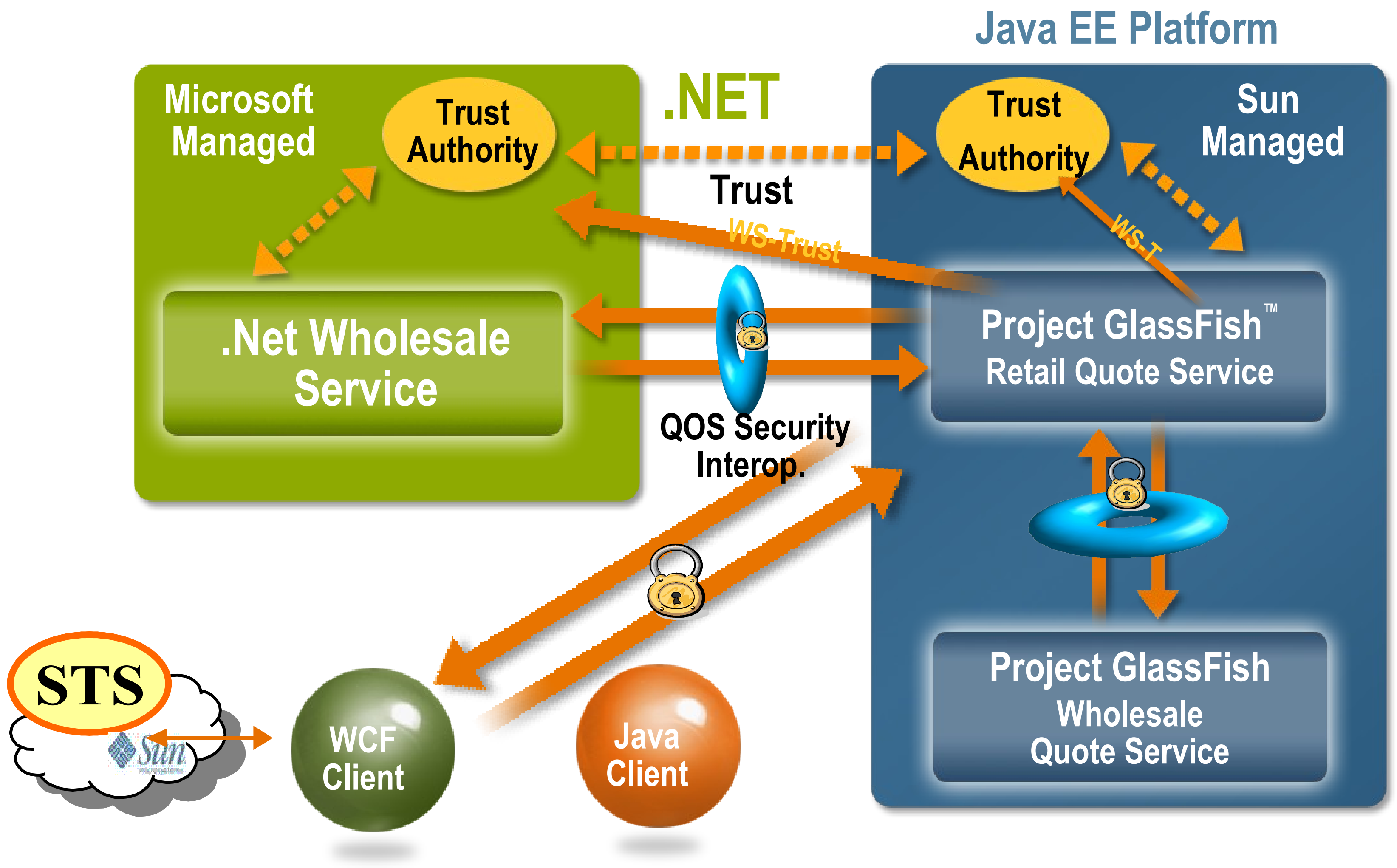
Trust

- WS-Trust framework for:
  - > Issue, Validate security tokens used by WS-Security
  - > Establish and broker trust relationships



```
<S11:Envelope xmlns:S11="..."
xmlns:wsse="...">
<S11:Header>
<wsse:Security>
.Security Token =
AuthN+AuthZ+signatures
</wsse:Security>
</S11:Header>
 <S11:Body wsu:Id="MsgBody">
   .. App data
  </S11:Body>
</S11:Envelope>
```

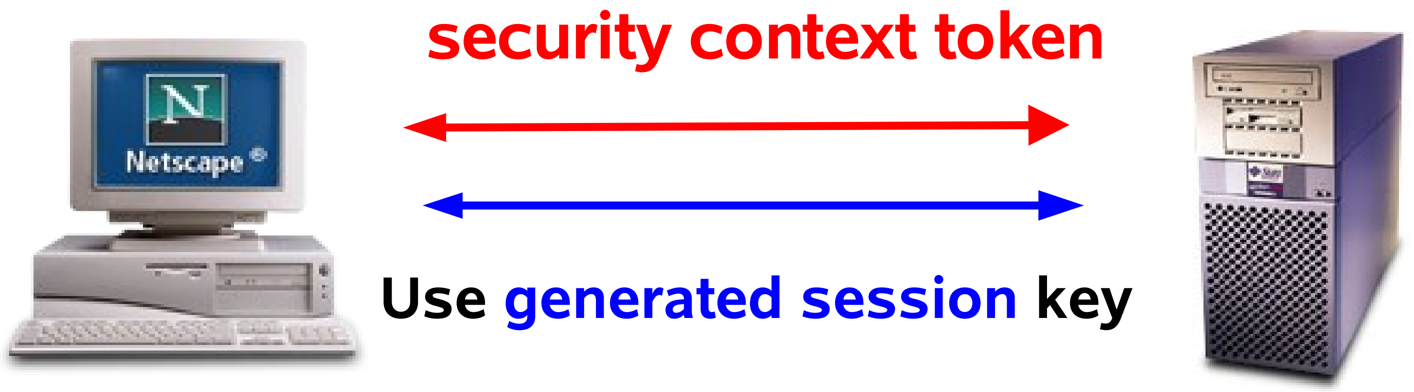
With Project Tango



WS-SecureConversation

Optimized Security

- How to Establish a Secure SESSION
  - > For multiple message exchanges
  - > Create shared symmetric session key
  - > Optimizes processing



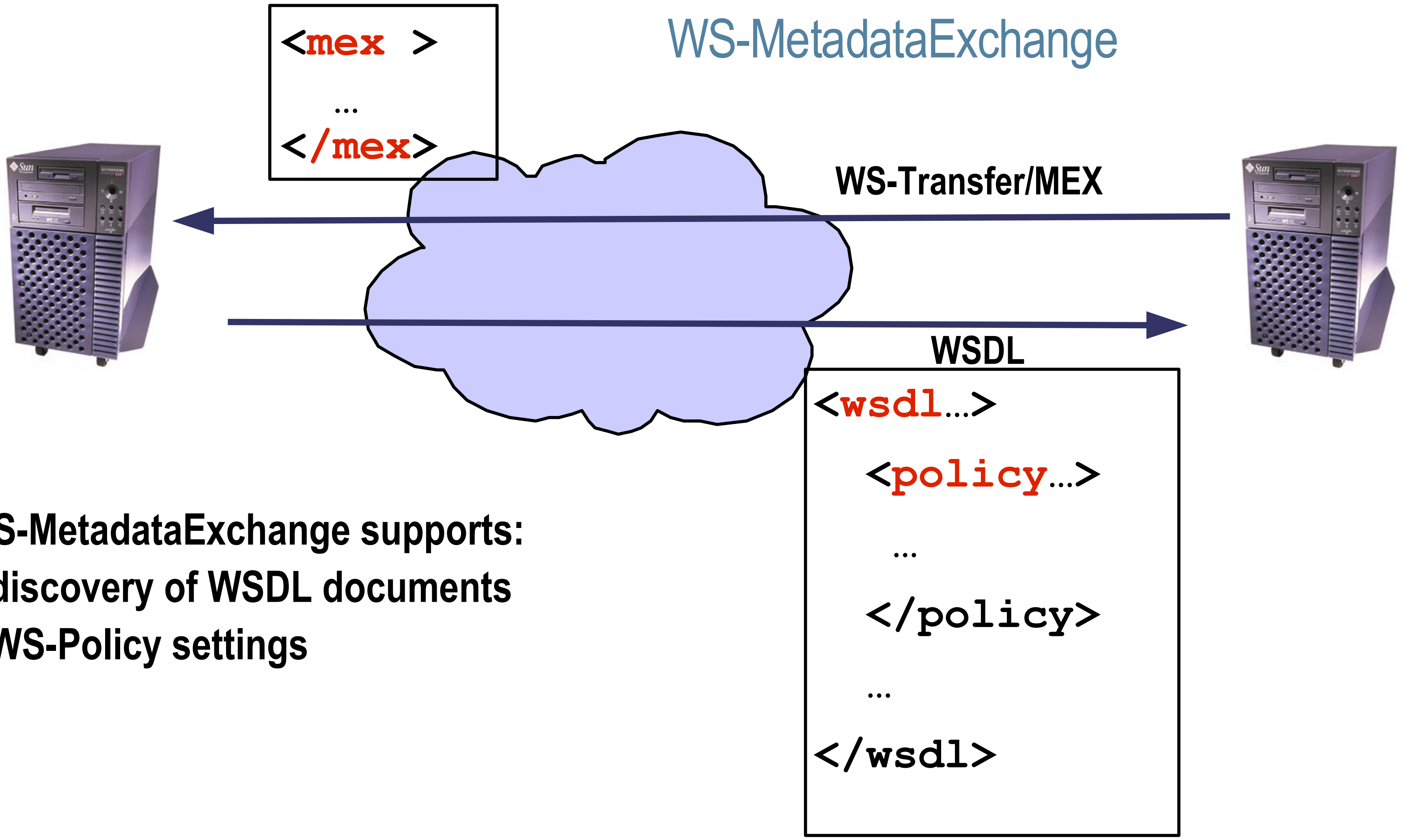
WS-Policy

```
<wsdl…>
  <policy…>
   …
  </policy>
  …
</wsdl>
```

```
<wsdl…>
  <policy…>
    <security-policy>
      …
    </security-policy>
    <transaction-policy>
      …
    </transaction-policy>
    <reliability-policy>
      …
    </reliability-policy>
   …
  </policy>
  …
</wsdl>
```

Metro: Bootstrapping

WS-Metadata Exchange
Bootstrapping Communication



```
<mex >
   …
</mex>
```

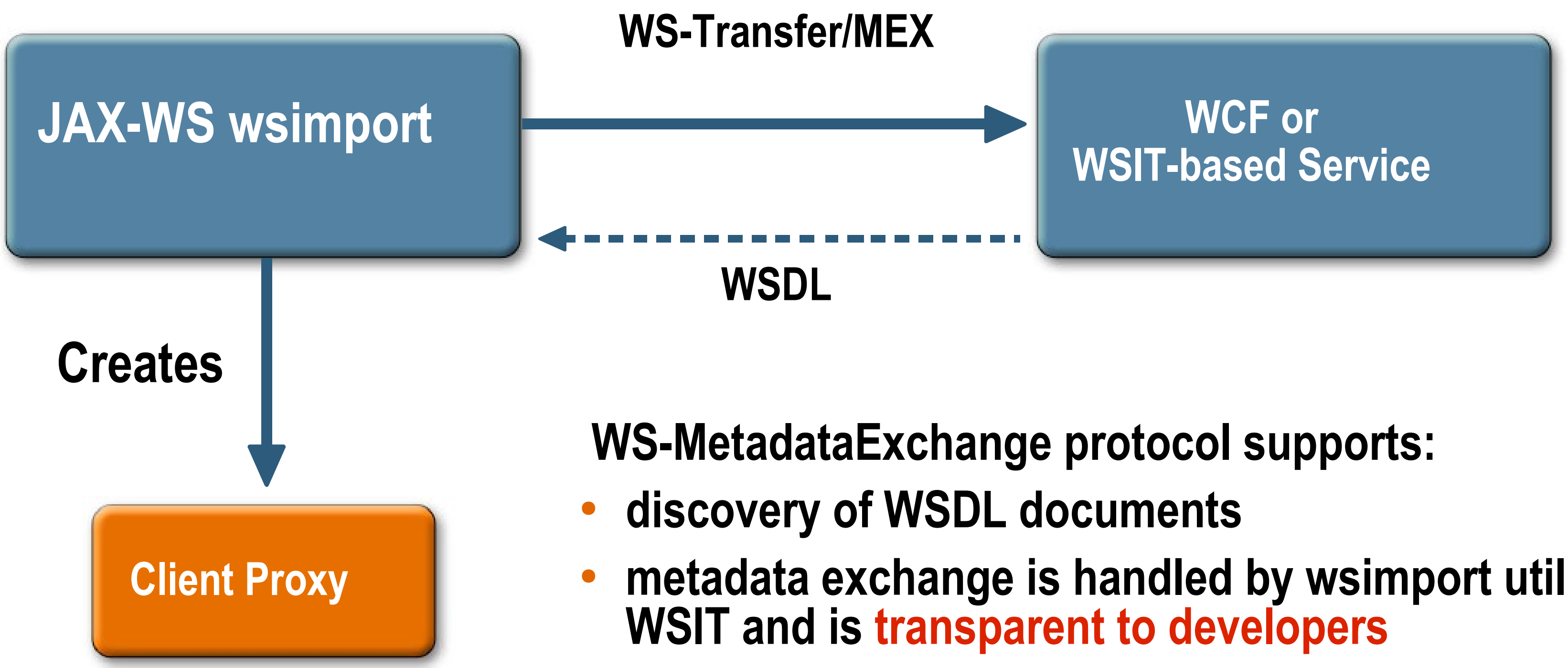
```
<wsdl…>
   <policy…>
    …
   </policy>
   …
</wsdl>
```

WS-MetadataExchange supports:

- discovery of WSDL documents
- WS-Policy settings

Bootstrapping Communication

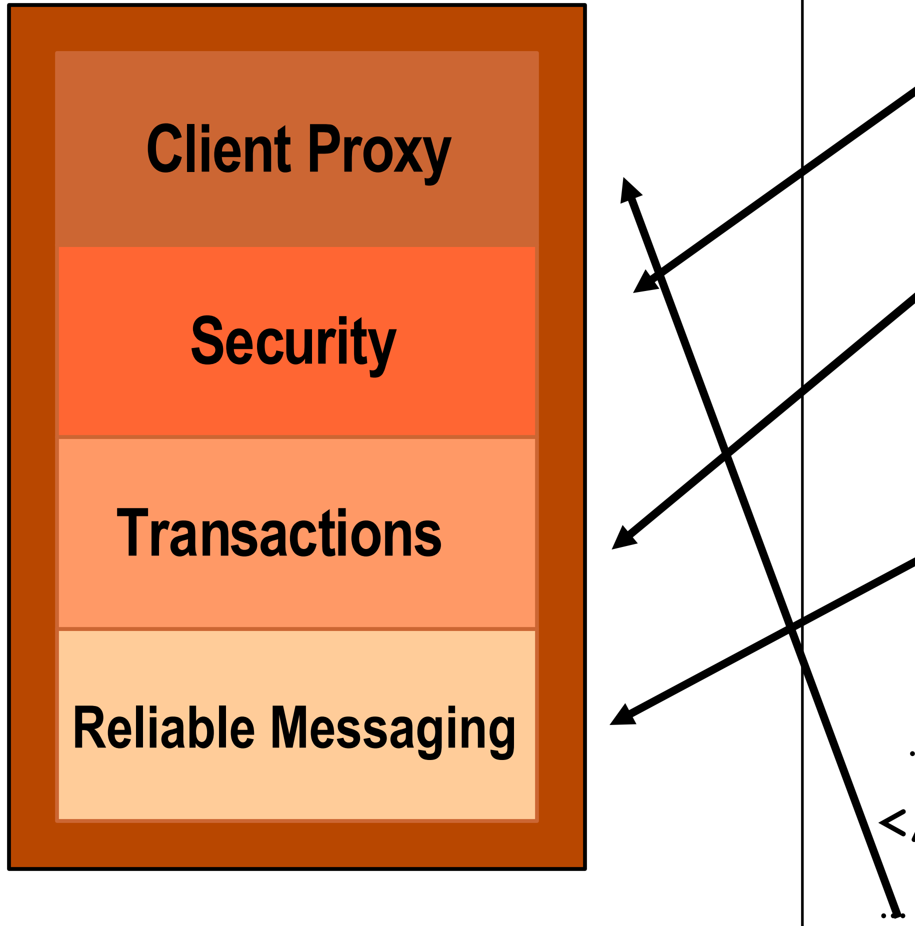
WS-MetadataExchange



WS-MetadataExchange protocol supports:

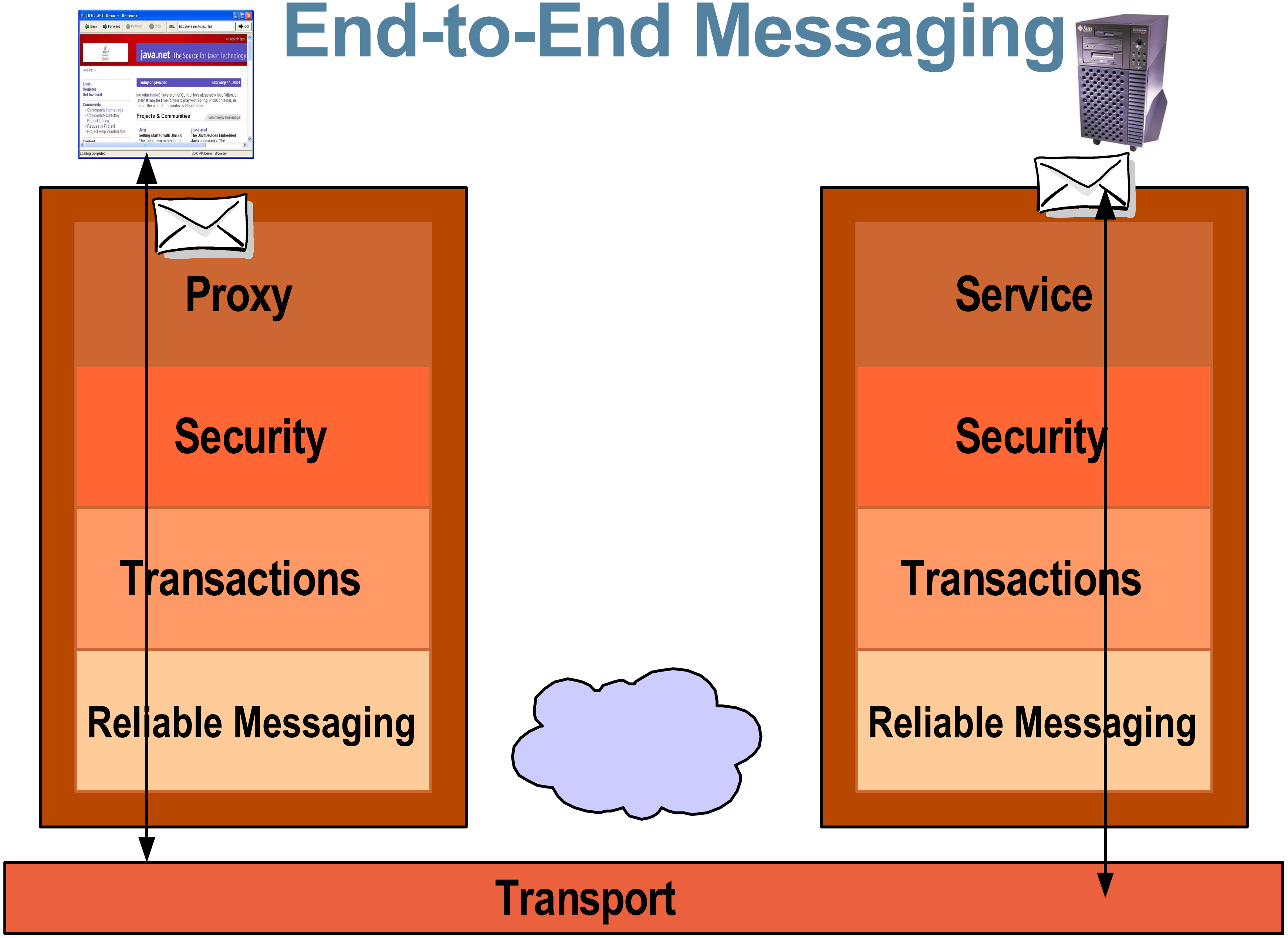
- discovery of WSDL documents
- metadata exchange is handled by wsimport utility of WSIT and is transparent to developers

Bootstrapping Communication Proxy Generation



```
<wsdl…>
  <policy…>
    <security-policy>
      …
    </security-policy>
    <transaction-policy>
      …
    </transaction-policy>
    <reliability-policy>
      …
    </reliability-policy>
   …
  </policy>
  …
</wsdl>
```

End-to-End Messaging

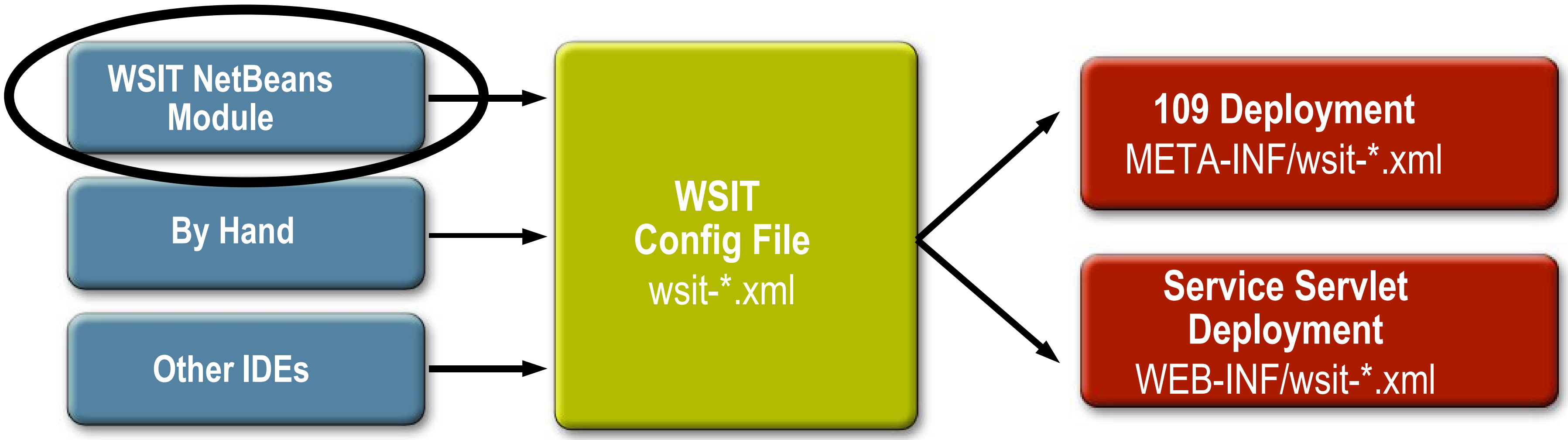


WSIT (Project Tango) Programming Model

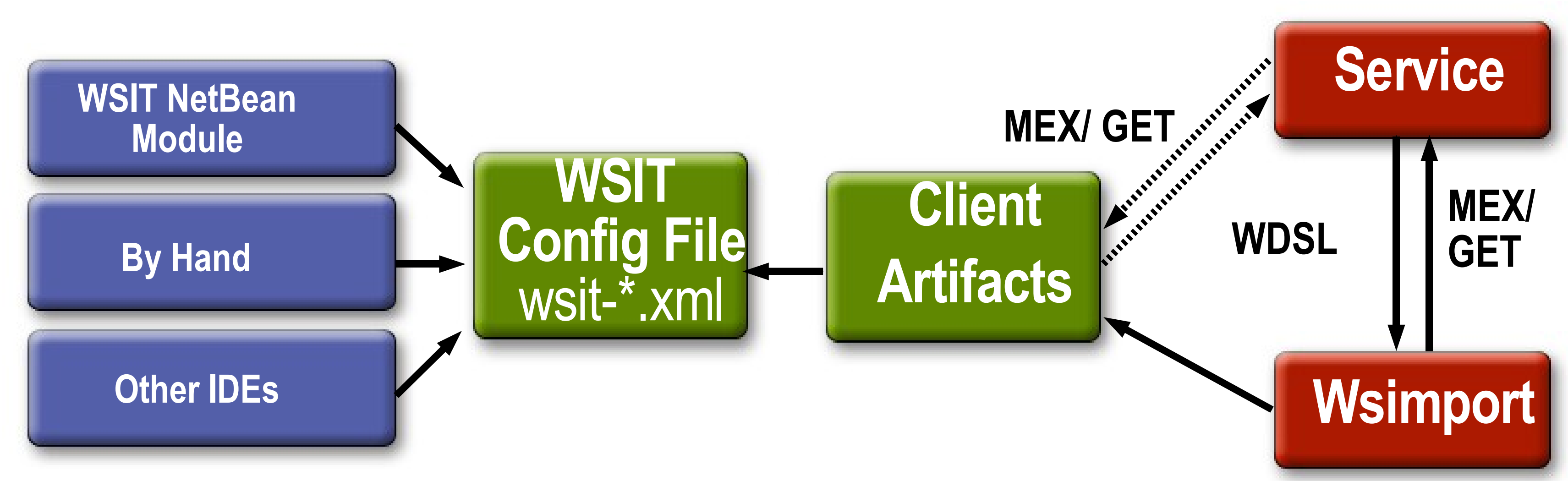
- **No runtime APIs** for WSIT
- Use **JAX-WS** and **EJB** APIs
- Developer/deployer supplies **config file** to enable/control Project Tango components
- Config file written by hand or produced by Project Tango NetBeans software module

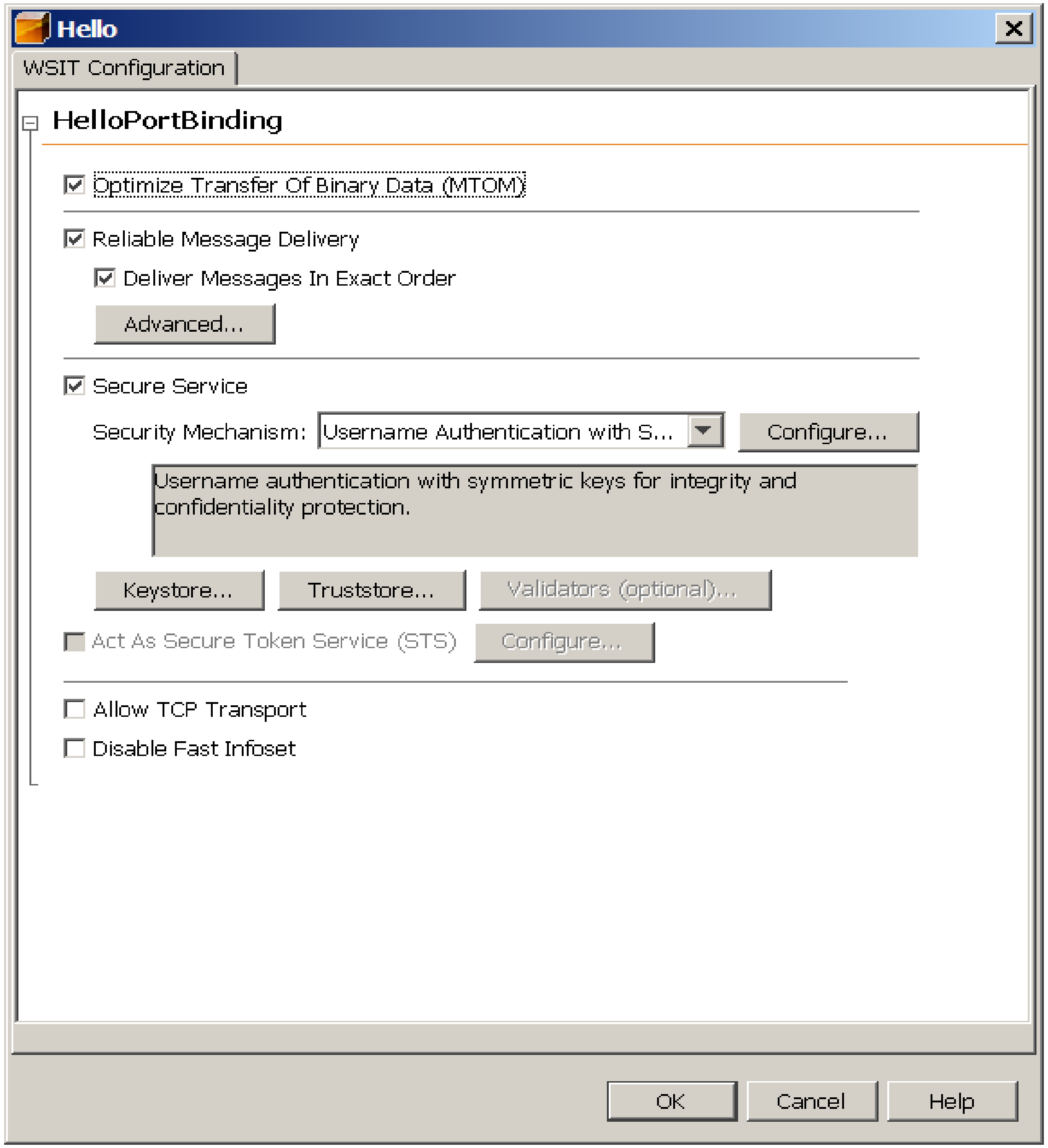
WSIT Server-Side Programming Model

- **No runtime APIs** for WSIT
- Use **JAX-WS** and **EJB** APIs
- Config file written by hand or produced by NetBeans enable/control WSIT


WSIT NetBeans Module
By Hand
Other IDEs
WSIT Config File wsit-*.xml
109 Deployment META-INF/wsit-*.xml
Service Servlet Deployment WEB-INF/wsit-*.xml

WSIT Client Programming Model





Hello

WSIT Configuration

HelloPortBinding

[✓] Optimize Transfer Of Binary Data (MTOM)

[✓] Reliable Message Delivery

[✓] Deliver Messages In Exact Order

Advanced...

[✓] Secure Service

Security Mechanism: Username Authentication with S...

Configure...

Username authentication with symmetric keys for integrity and confidentiality protection.

Keystore... Truststore... Validators (optional)...

[ ] Act As Secure Token Service (STS) Configure...

[ ] Allow TCP Transport

[ ] Disable Fast Infoset

OK Cancel Help

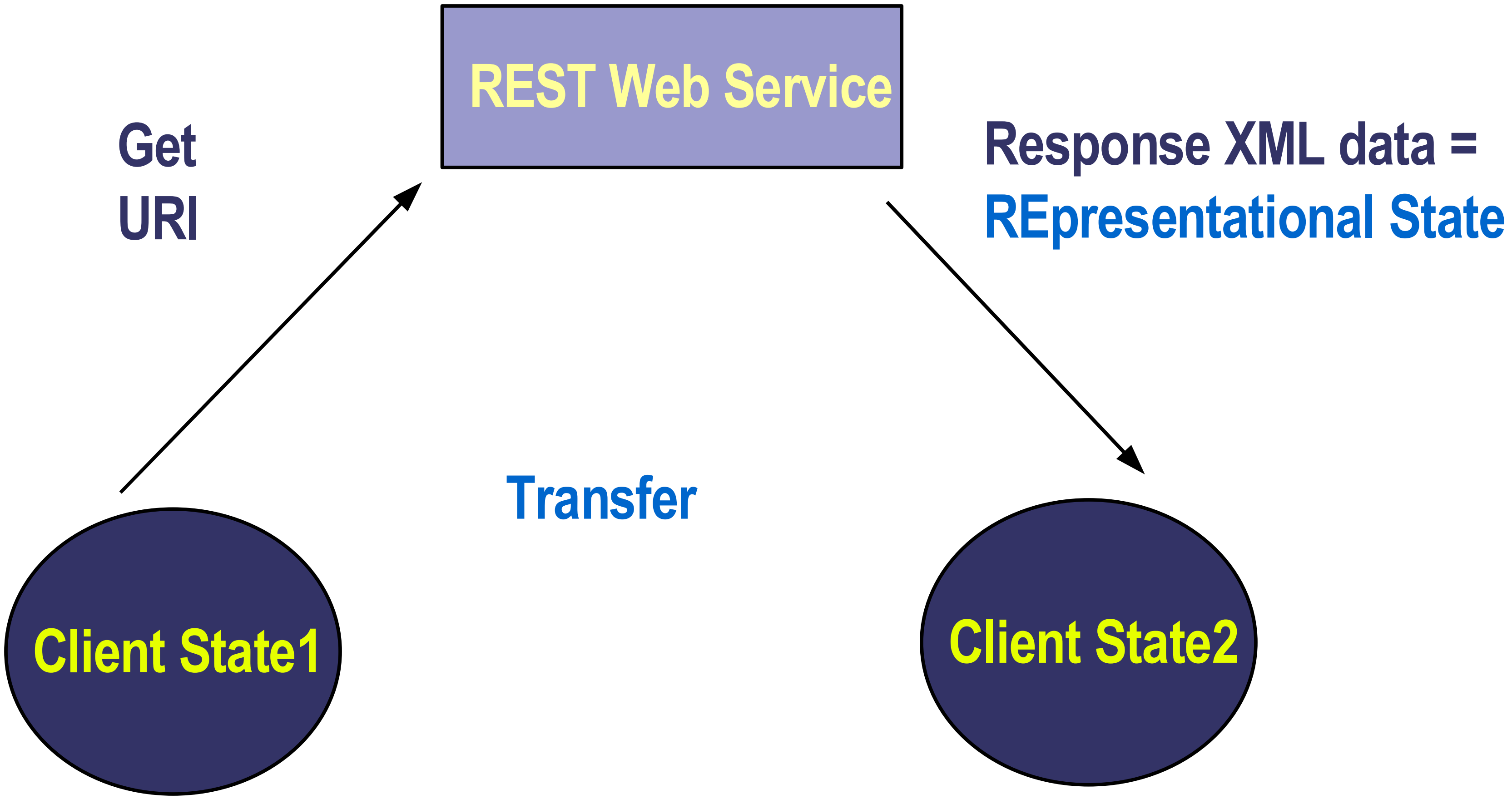
Agenda

- Metro
- JAX-WS Standards
- WSIT
- REST with JAX-RS

API: JAX-RS

- Standardized in the JCP
  - JSR 311
  - Will be included in Java EE 6
- EG members
  - Alcatel-Lucent, BEA, Day Software, Fujitsu, innoQ, Nortel, Red Hat
  - Experts in Atom, AtomPub, WebDAV, HTTP, REST, Restlet
- Group started in April

REpresentational State Transfer



REST Tenets

- **RE**presentational **S**tate **T**ransfer
- Resources (nouns)
  - Identified by a URI, For example:
    - http://www.parts-depot.com/parts
- Methods (verbs)
  - Small fixed set:
    - Create, Read, Update, Delete
- State Representations
  - data and state transferred between client and server
  - XML, JSON...

HTTP Example

Method

Request

```
GET /music/artists/beatles/recordings HTTP/1.1
Host: media.example.com
Accept: application/xml
```

Resource

Response

```
HTTP/1.1 200 OK
Date: Tue, 08 May 2007 16:41:58 GMT
Server: Apache/1.3.6
Content-Type: application/xml; charset=UTF-8

<?xml version="1.0"?>
<recordings xmlns="…">
  <recording>…</recording>
  …
</recordings>
```

Representation

CRUD to HTTP method mapping

CRUD methods	4 main HTTP methods	
	Verb	**Noun**
Create	POST	Collection URI
Read	GET	Collection URI
Read	GET	Entry URI
Update	PUT	Entry URI
Delete	DELETE	Entry URI

Example

- Music Collection
  - `/music/artists/{id}`
  - `/music/artists/{id}/recordings`

- URI Templates are URIs with variables within the URI syntax.

Artist Resource Using Servlet API

Don't try to read this,
this is just to show the complexity
:)

```
public class Artist extends HttpServlet {

    public enum SupportedOutputFormat {XML, JSON};

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String accept = request.getHeader("accept").toLowerCase();
        String acceptableTypes[] = accept.split(",");
        SupportedOutputFormat outputType = null;
        for (String acceptableType: acceptableTypes) {
            if (acceptableType.contains("*/*") || acceptableType.contains("application/*") ||
                acceptableType.contains("application/xml")) {
                outputType=SupportedOutputFormat.XML;
                break;
            } else if (acceptableType.contains("application/json")) {
                outputType=SupportedOutputFormat.JSON;
                break;
            }
        }
        if (outputType==null)
            response.sendError(415);
        String path = request.getPathInfo();
        String pathSegments[] = path.split("/");
        String artist = pathSegments[1];
        if (pathSegments.length < 2 && pathSegments.length > 3)
            response.sendError(404);
        else if (pathSegments.length == 3 && pathSegments[2].equals("recordings")) {
            if (outputType == SupportedOutputFormat.XML)
                writeRecordingsForArtistAsXml(response, artist);
            else
                writeRecordingsForArtistAsJson(response, artist);
        } else {
            if (outputType == SupportedOutputFormat.XML)
                writeArtistAsXml(response, artist);
            else
                writeArtistAsJson(response, artist);
        }
    }
    private void writeRecordingsForArtistAsXml(HttpServletResponse response, String artist) { ... }

    private void writeRecordingsForArtistAsJson(HttpServletResponse response, String artist) { ... }

    private void writeArtistAsXml(HttpServletResponse response, String artist) { ... }

    private void writeArtistAsJson(HttpServletResponse response, String artist) { ... }
}
```

JAX-RS = Easier REST Way

Server-side API Wish List

- High level, Declarative
  - Uses @ annotation in POJOs

- **Disclaimer**: Early in the Java Specification Request (JSR) process, everything from here on in **liable to change**!

Clear mapping to REST concepts

- Resources: what are the URIs?

```
@UriTemplate("/artists/{id}")
```

- Methods: what are the HTTP methods?

```
@HttpMethod("GET")
public XXX find()
```

- Representations: what are the formats?

```
@ConsumeMime("application/xml")
@ProduceMime("application/json")

 (New types can be defined)
```

POJO

responds to the URI http://host/music/artists/{id}

responds with XML

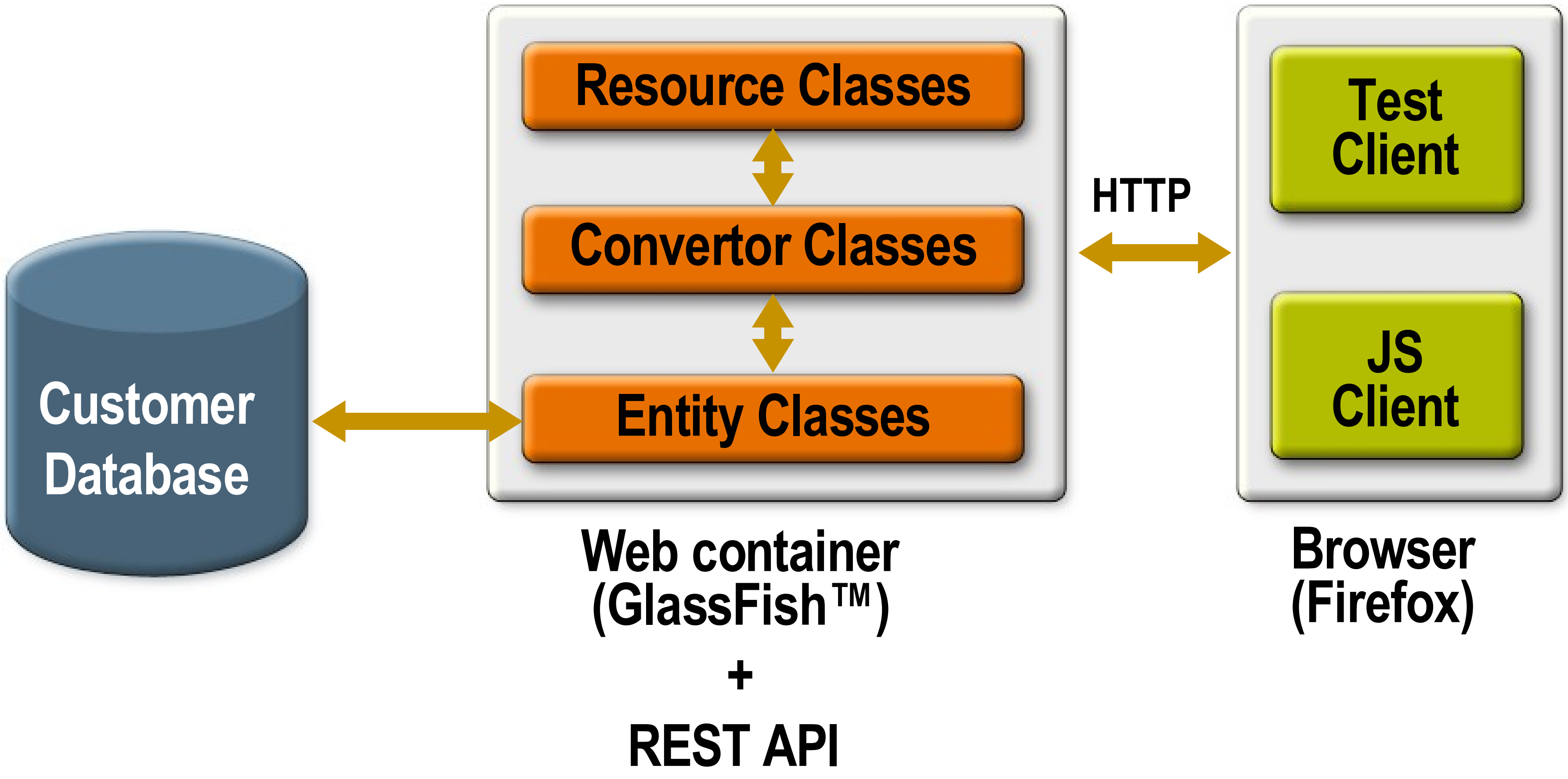
responds to HTTP GET

```
@UriTemplate("/artists/{id}")
public class Artist {

    @ProduceMime("application/xml")
    @HttpMethod("GET")
    InputStream getXml(
      @UriParam("id") String artist) {
        ...
    }


    ...
}
```

Artist Resource as JAX-RS POJO

```
@UriTemplate("/artists/{id}")
@ProduceMime("application/xml")
public class Artist {
    @HttpMethod
    InputStream getXml(@UriParam("id") String artist) { ... }

    @HttpMethod
    @ProduceMime("application/json")
    InputStream getJson(@UriParam("id") String artist) { ... }

    @HttpMethod
    @UriTemplate("recordings")
    InputStream getRecordingsXml(@UriParam("id") String artist) { ... }

    @HttpMethod
    @ProduceMime("application/json")
    @UriTemplate("recordings")
    InputStream getRecordingsJson(@UriParam("id") String artist) { ... }
}
```

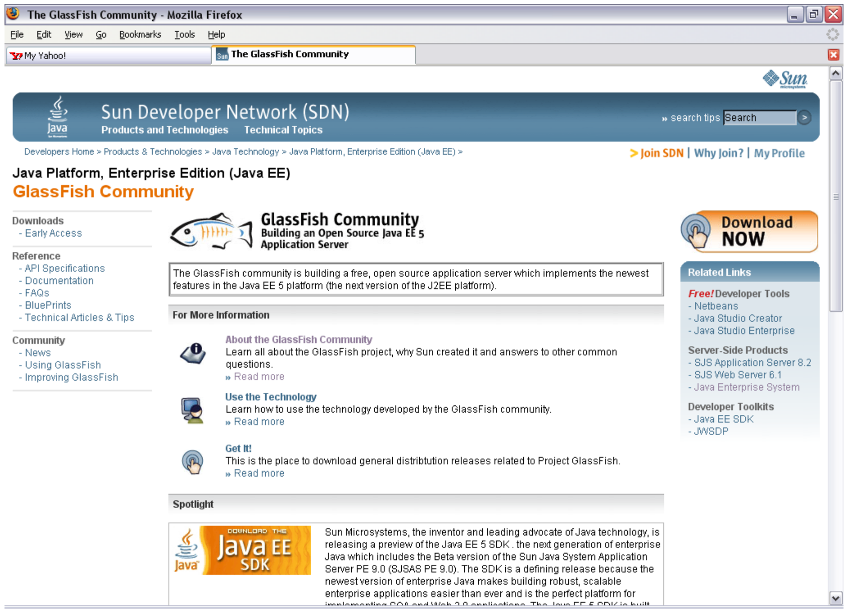
Customer Service Overview



Summary

- Metro Integrated with GlassFish Application Server
  - JAX-WS
    - easier to use and more powerful than JAX-RPC
    - part of the Java EE 5 and Java SE 6 platforms
    - Layered design hides the complexity
      - Extensible at the protocol and transport level
  - WSIT
    - Makes Metro interoperable with other WS-* stacks
    - No new APIs , easy with NetBeans plugin
- JAX-RS
  - High-level declarative programming model for REST

Project GlassFish


The GlassFish Community - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
My Yahoo!
The GlassFish Community
Sun Developer Network (SDN)
Products and Technologies
Technical Topics
» search tips
Search
> Join SDN | Why Join? | My Profile
Download NOW

Developers Home > Products & Technologies > Java Technology > Java Platform, Enterprise Edition (Java EE) >

Java Platform, Enterprise Edition (Java EE)

GlassFish Community

Downloads
- Early Access

Reference
- API Specifications
- Documentation
- FAQs
- BluePrints
- Technical Articles & Tips

Community
- News
- Using GlassFish
- Improving GlassFish

GlassFish Community
Building an Open Source Java EE 5 Application Server

The GlassFish community is building a free, open source application server which implements the newest features in the Java EE 5 platform (the next version of the J2EE platform).

For More Information

About the GlassFish Community
Learn all about the GlassFish project, why Sun created it and answers to other common questions.
» Read more

Use the Technology
Learn how to use the technology developed by the GlassFish community.
» Read more

Get It!
This is the place to download general distribtution releases related to Project GlassFish.
» Read more

Spotlight

DOWNLOAD THE Java EE SDK

Sun Microsystems, the inventor and leading advocate of Java technology, is releasing a preview of the Java EE 5 SDK . the next generation of enterprise Java which includes the Beta version of the Sun Java System Application Server PE 9.0 (SJSAS PE 9.0). The SDK is a defining release because the newest version of enterprise Java makes building robust, scalable enterprise applications easier than ever and is the perfect platform for

Related Links

Free! Developer Tools
- Netbeans
- Java Studio Creator
- Java Studio Enterprise

Server-Side Products
- SJS Application Server 8.2
- SJS Web Server 6.1
- Java Enterprise System

Developer Toolkits
- Java EE SDK
- JWSDP

Building a Java EE 5 Open Source Application Server

Simplifying Java application Development with Java EE 5 technologies

Includes JWSDP, EJB 3.0, JSF 1.2, JAX-WS and JAX-B 2.0

Supports > 20 frameworks and apps

Basis for the Sun Java System Application Server PE 9

Free to download and free to deploy

Over 1200 members and 200,000 downloads

Integrated with NetBeans

java.sun.com/javaee/GlassFish

Source: Sun 2/06—See website for latest stats

For More Information

- METRO
  - http://metro.dev.java.net
- JAX-WS
  - http://jax-ws.dev.java.net
- WSIT
  - http://wsit.dev.java.net
- REST
  - http://jersey.dev.java.net
- Glassfish
  - http://glassfish.dev.java.net

Metro: JAX-WS, WSIT and REST

Carol McDonald
carol.mcdonald@sun.com