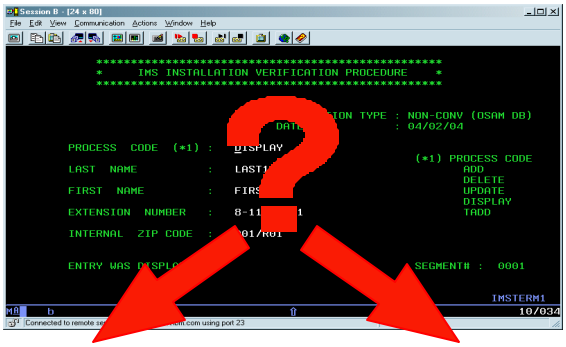
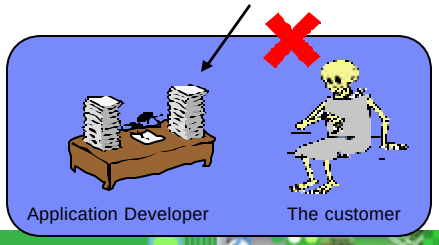
Agenda

- **J2EE and JCA Connector Architecture**
- **IMS Connector for Java**
  - Overview
  - Features / Enhancements
- **Developing Applications to Access IMS Transaction**
- **Summary**

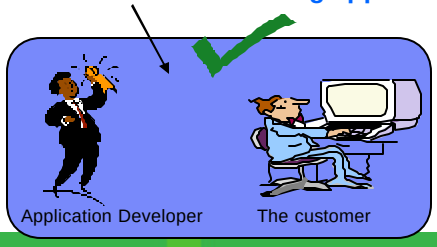


Rewrite existing apps for the web

Web-enable existing apps



OR

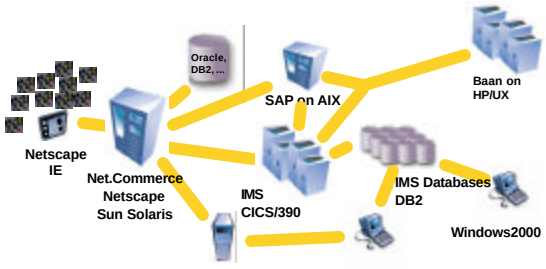


e-Business Applications

- **An e-Business application is made up of components that can run in one or more physical computers**
- **Using Java for such an application removes portability issues**
  - deployment platform can be determined by where the data is or by capacity requirements, etc.

- **But such applications require system services such as -**
  - security
  - access to legacy systems
  - distributed syncpointing

J2EE

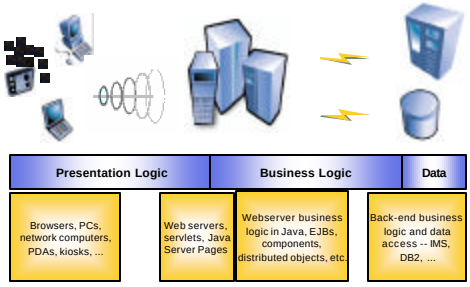


J2EE

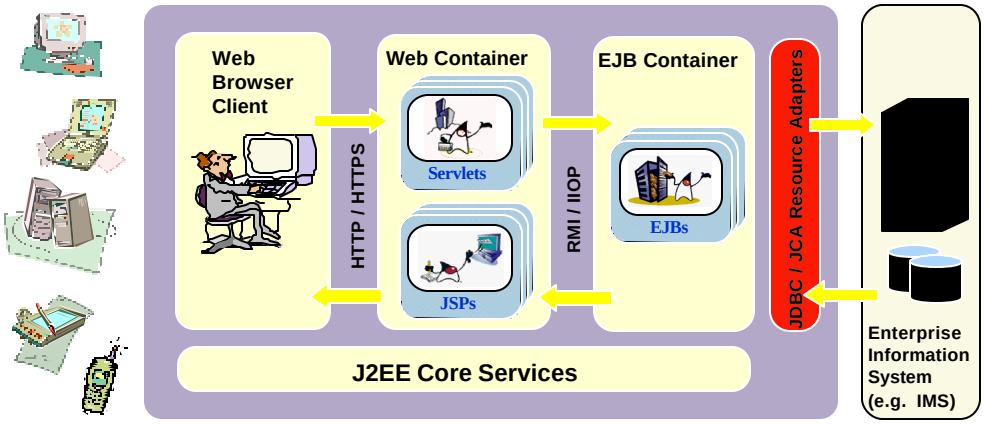
- Server model
- Connector architecture

J2EE Architecture

- **Java 2 Platform, Enterprise Edition**
  - A standards -based architecture to enable development of multi-tier distributed applications
  - Recognizes long term dependency on existing legacy systems
    - "Enterprise Information Systems (EIS)"
  - The business logic in the middle tier runs as Enterprise Java Beans (EJB)
    - EJBs can call other EJBs (anywhere) or EIS systems



J2EE Server Model

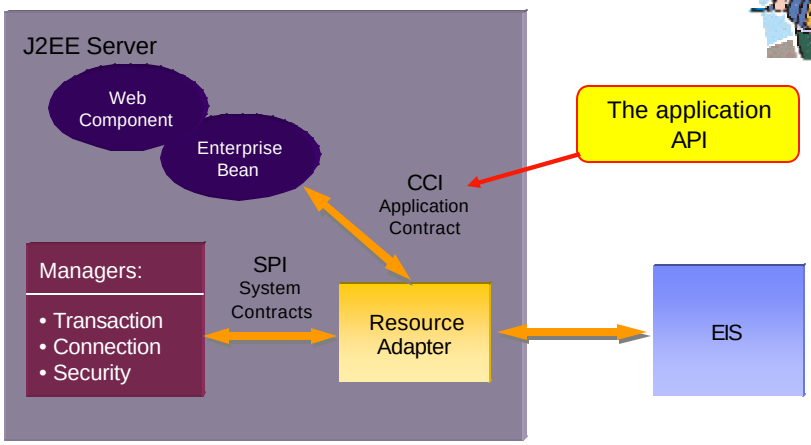


Doing business on the Web continues to be a top priority for companies worldwide. Many are turning to Java 2 platform Enterprise Edition (J2EE) technology as the foundation for their web-based applications. META estimates J2EE is utilized by at least 80% of the Global 2000.

J2EE Connector Architecture (JCA)

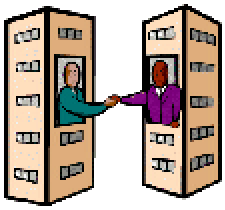
- **An extension of J2EE architecture**
- **Defines a common set of interfaces - Service Provider Interface (SPI) and Common Client Interface (CCI) - that standardizes interactions between:**
  - Resource Adapters
  - Application Servers
  - Application Components
- **CCI is the standard API to be used for all application access to any EIS**
  - e.g.. EJB access to an IMS Transaction
- **No longer need to customize a connector for each application server, and vice versa**

J2EE Connector Architecture



Quality of Service (QoS)

- **J2EE Connector Architecture defines how the application server and EIS Resource Adapter interact to manage quality of services**
  - Connection Management
    - Connection Pooling and reuse
  - Security Management
    - EIS Sign-On
  - Transaction Management
    - Global Transaction with Two Phase Commit processing
  - Tracing and Logging



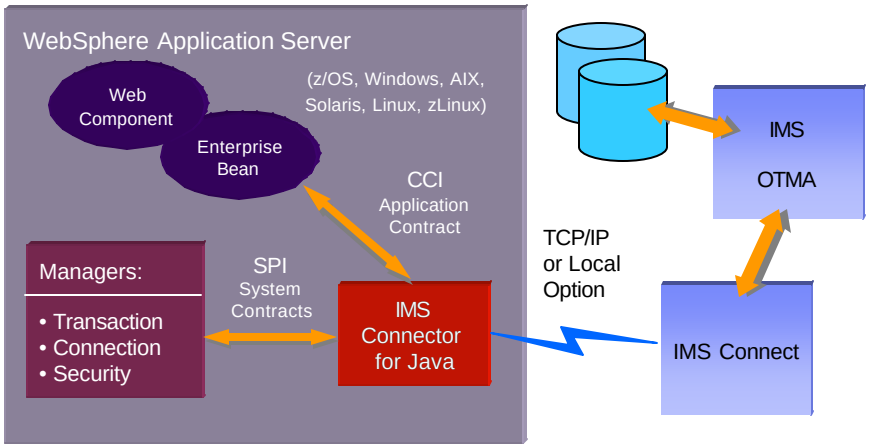
IMS Connector for Java

IMS Connector for Java

- **A J2EE Connector Architecture (JCA) resource adapter**
- **Used to develop and run J2EE application that access IMS transactions via IMS Connect**
- **Offers a highly scalable and flexible topology**
- **Supports rapid application development with WebSphere Studio Application Developer Integration Edition**
- **Runs in WebSphere Application Server on both z/OS and distributed platforms**

"Enables IMS Users to integrate their core business transactions into their On Demand world"

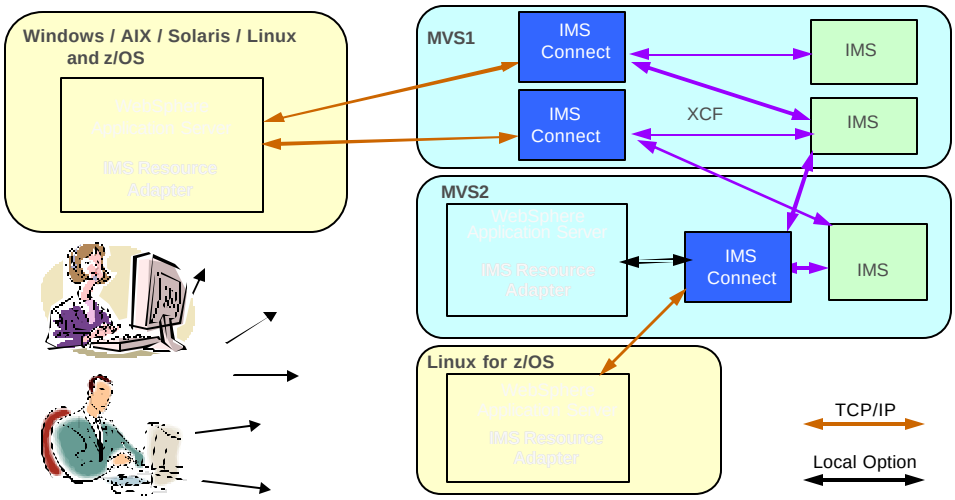
J2EE Connector Architecture with IMS



Key Functions of IMS Resource Adapter

- **The application (Servlet/EJB) –**
  - creates an Input Message Bean (java object) and puts values into the input variables
  - calls a Format Handler to convert the bean into an IMS input message (byte array) and passes it to the IMS Resource Adapter
- **The IMS Resource Adapter –**
  - Receives the IMS input byte array
  - Builds the OTMA header using parameters passed by the application
  - Adds the IMS Connect header (IRM)
  - Uses TCP/IP socket calls (or local option) to communicate with IMS Connect and IMS
    - Interfaces with Container for connection management and security
  - Receives reply from IMS and IMS Connect
  - Removes headers
  - Returns reply message byte array to application
- **The application –**
  - Receives reply byte array and calls Format Handler to convert it to an Output Message Bean

IMS Resource Adapter – *Flexible and Scalable Topology*



Features of IMS Connector for Java

IMS Connector for Java Features

- **TCP/IP and Local Option connection**
- **IMS conversational and non-conversational transactions**
- **Send-then-commit (Commit Mode 1) with persistent sockets**
- **Local z/OS RRS 2-phase commit**
- **Container and Component Managed Sign-on**

New in 2003!

- **Distributed XA 2-phase commit**
- **Message Format Service (MFS)**
- **Secure Socket Layer (SSL)**
- **Commit-then-send (Commit Mode 0)**
  - Non-persistent socket
- **Retrieve asynchronous output messages (CM0)**
- **Execution timeout**
  - Require IMS Connect 2.1 and IMS V8

IMS Connector for Java Features - 2004

New in 2004!

- **Socket Timeout**
- **Connection Retry**
- **CCI Record Helper class**
  - **Conversational sample application**
- **Persistent Sockets for CM0**
- **SEND_ONLY transaction input***

IMS Connector for Java

- **Version 2.2**
  - GA June 2004
- Supplied with WSADIE V5.1
- Requires
  - IMS Connect 2.1 with PTF or *IMS Connect 2.2
  - IMS V8/9 with PTF
  - WAS with PTF

Timeouts

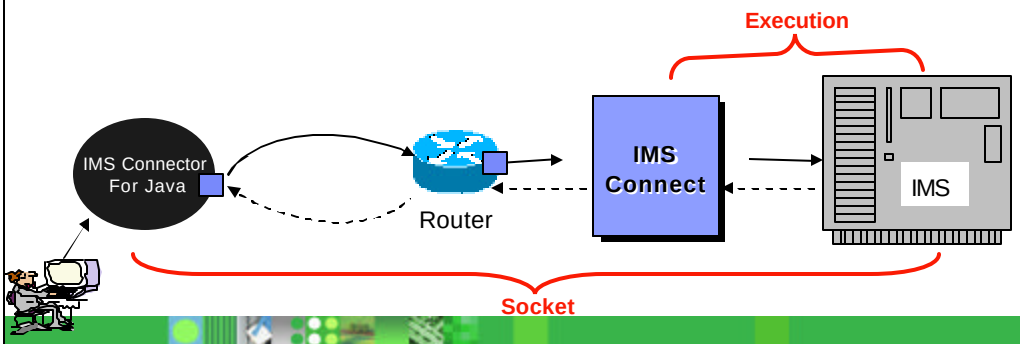
New

Execution timeout

- Represents how long IMS Connect should wait for IMS to return the output of a transaction.
- IMS hangs
- Set per interaction

Socket Timeout

- Represents how long IMS Resource Adapter should wait for IMS Connect to return.
- Network problems
- Set per interaction



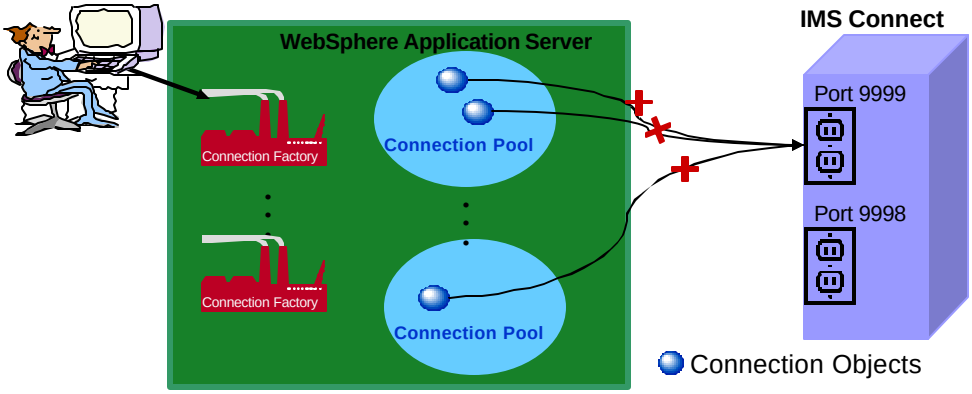
Connection Management

- **Connection Pooling**
  - WebSphere Application Server maintains pools of connections
    - A pool is defined for each Connection Factory (hostName/port/dataStore combination)
  - Unused connection objects are returned to the pool for re-use
    - The associated TCP/IP socket or Local Option connection remains open
- **Connection Management Properties**
  - MaxConnection, MinConnection, ReapTime, UnusedTimeout, ConnectionTimeout and Purge Policy
    - Purge Policy specifies how to purge connections when a communication error is detected.
      - Valid values are EntirePool and FailingConnectionOnly

Connection Retry

New

- When IMS Connect is recycled
  - All connections in pool become unusable
  - All connections in pool would be tried before new connections are created
- **Solution: IMS Connector for Java attempts to restore the connection**



Security Management

- **End-to-end secure access of an EIS by J2EE applications**
- **The IMS Resource Adapter supports**
  - **User ID and Password authentication mechanism**
  - **Already verified security identity when running in WAS z/OS with Local Option**
    - Security information is supplied to the IMS Resource Adapter by:
      - The application component (Component-managed Sign-on)
        or
      - The application server (Container-managed Sign-on)

Application deployment option

    - IMS Resource Adapter passes the security information
      - To IMS Connect, as required, for authentication
      - To IMS OTMA for authorization
  - **Encryption using SSL**
    - **Provides reliable, secure communication between IMS Resource Adapter and IMS Connect**

Transaction Management

- **Global transaction 2-phase commit for distributed environments**
  - Enables you to make consistent changes to one or more protected resources in a single unit of work (a transaction) such that all changes are either fully completed or fully rolled back
  - Implements XA protocol to support the coordination of changes in distributed resources
  - RRS transaction with Local Option in WAS z/OS

- **WebSphere Application Server**
  - Transaction manager
  - Supports 2 phase commit protocol
- **J2EE offers two options on how to set the transaction boundaries in your application**
  - Bean-Managed Transaction
  - Container-Managed Transaction

Application coding or deployment options

Commit Mode and Persistent Socket Support

- **Application specifies Commit Mode (0 or 1)**
- **IMS Connector for Java always uses Persistent Sockets**
  - Two types – **Sharable** or **Dedicated**
    - CM1 uses Sharable
    - Application specifies type for CM0

New

	ClientID	IMSInteractionVerb	Socket	SyncLevel
CM1	Generated by IMS Resource Adapter	SYNC_SEND_RECEIVE	Sharable Persistent	NONE
CM0	Generated by IMS Resource Adapter	SYNC_SEND_RECEIVE SYNC_SEND	Sharable Persistent	CONFIRM
CM0	User-Specified ClientID is REQUIRED	SYNC_SEND_RECEIVE SYNC_SEND SYNC_RECEIVE_ASYNCOUTPUT	Dedicated Persistent	CONFIRM

Developing Applications with IMS Connector for Java

- Common Client Interface (CCI)
- WebSphere Studio Application Developer Integration Edition (WSADIE)

Developing Applications

Java Application

Enterprise JavaBean

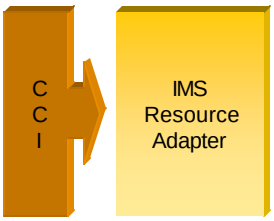
Web Application

Web Service

- **Build application code using the Common Client Interface to interact with the IMS Resource Adapter**

or

- **Use *WebSphere Studio Application Developer Integration Edition* to rapidly generate the application code**



(The generated code uses the CCI to interact with the Resource Adapter)

Using the Common Client Interface (CCI)

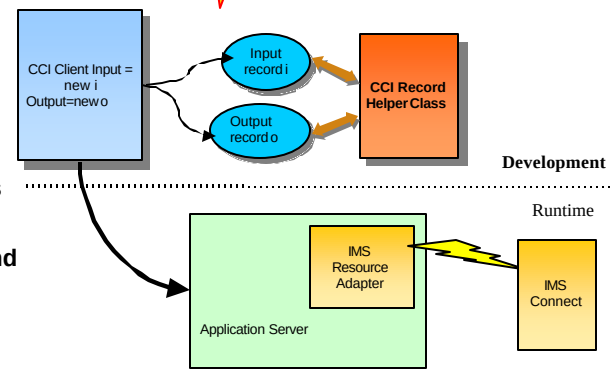
```
Context ic = new InitialContext();   // Establish a JNDI context

// Use JNDI to look up a ConnectionFactory for the EIS (IMS)
ConnectionFactory cf = (ConnectionFactory) ic.lookup("java:comp/env/eis/myIMS");
Connection conn = cf.getConnection();   // Obtain a connection to the EIS (IMS)

// Create an interaction object to use with the EIS (IMS)
Interaction interaction = conn.getInteraction();

// Specify properties of the interaction
IMSInteractionSpec iSpec = new IMSInteractionSpec();
iSpec.setInteractionVerb(SYNC_SEND_RECEIVE);
...
// Create an input record and set the input values
// Create an output record
...
// Use the interaction object to invoke an EIS function (e.g., an
// IMS transaction)
interaction.execute(iSpec, input, output);

interaction.close();   // Close the interaction
conn.close();   // and connection
```

- **J2EE Connector Architecture defines a set of interfaces and classes used by the client to interact with an EIS**
- **The harder part is building the input bean and converting it to an IMS message byte array (and similarly for output)**

CCI Record Helper Class

New

- **Build Input and output message beans**
- **Convert to IMS byte-arrays**
- **Support Conversational and Non-conversational transaction access**



Can be used on any J2EE compliant platform

WSADIE and Enterprise Services

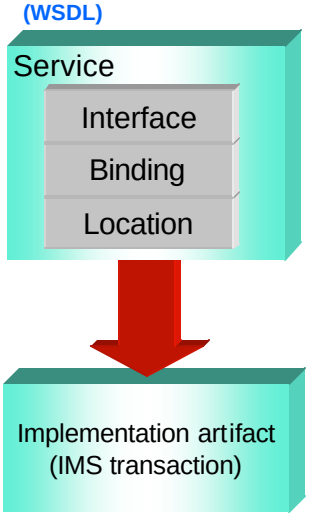
- **WSADIE is based on the concept of an "Enterprise Service"**
  - also referred to simply as a "Service"
- **For example, you use WSADIE to set up an IMS transaction as an Enterprise Service**

- **An Enterprise Service is not directly usable in production environments**
  - It needs to be "deployed" in a way that makes it usable
- **There are two key ways of *deploying* an IMS Enterprise Service**
  - As an EJB within a J2EE application
  - As a Web Service (or SOAP Service)
  - (It could be deployed as a Servlet. But it is no longer recommended to access IMS directly from a servlet, since there is a lack of "quality of service")

Enterprise Service

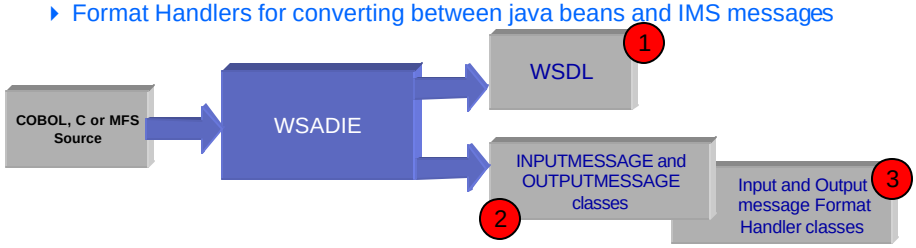
- **A service consists of:**
  - **An implementation artifact** (e.g., a Java class, EJB, IMS transaction, etc.)
  - **3 WSDL files that use XML to describe the service**:
    - Abstract Service Interface
      - Description of the operations and the messages they exchange
      - WSDL "portType"
    - Implementation Binding
      - Description of how service interface is physically implemented
    - Implementation Service
      - Location of the service

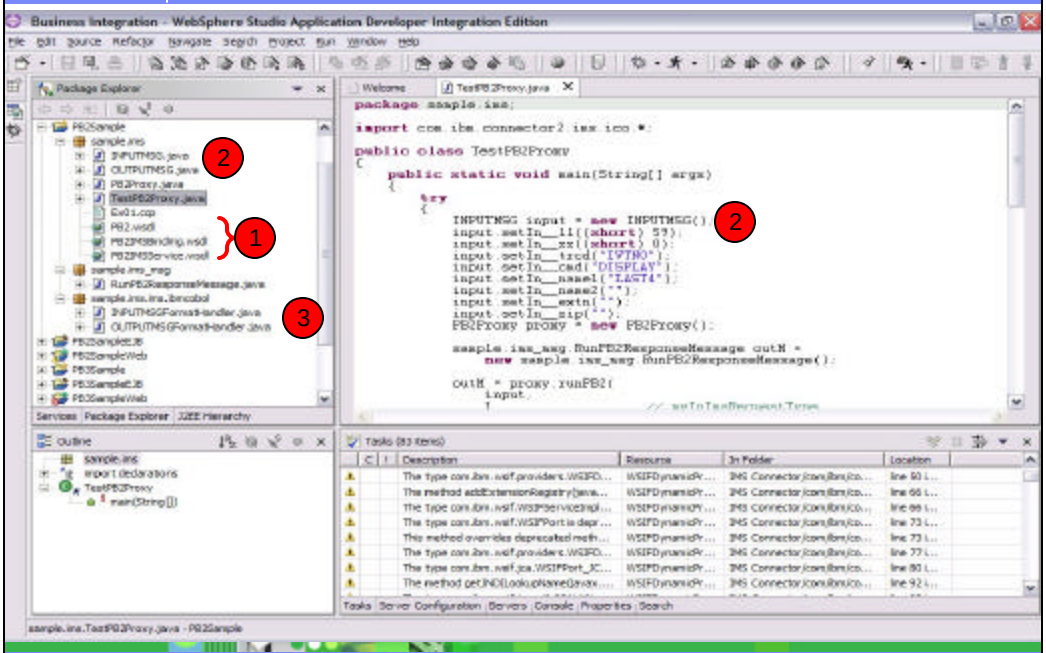
Web Services Description Language (WSDL)



Building the Service

- **Download the COBOL, C, or MFS source**
- **Use WSADIE to create an IMS Enterprise Service**
- **WSADIE creates the WSDL files that use XML to describe**
  - the fields in the input and output messages
  - how and where the IMS transaction is accessed (in test)
- **It also generates java classes**
  - to represent and build input and output messages (beans)
  - Format Handlers for converting between java beans and IMS messages



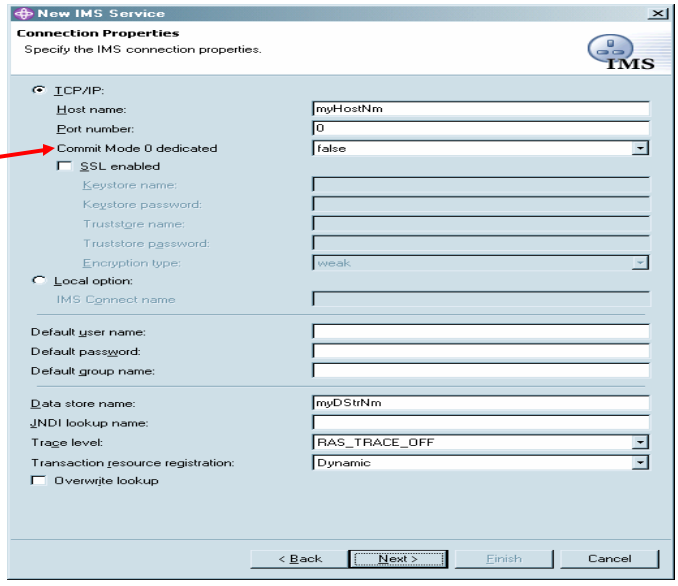


Building the Service

- **In the WSDL generation process, WSADIE will present you with a series of screens for specifying:**
  - **Connection Properties**
    - TCP Host name and Port number (for IMS Connect)
      - IMS Connect jobname if using Local Option
    - Commit Mode 0 type of persistent socket
    - Datastore name
    - Optional default security information (userid, groupid, password)
  - **Operation Binding Properties**
    - Commit Mode (0 or 1)
    - Request type (transaction, command or MFS-generated transaction)
    - Interaction type (send-and-receive, send-only, resume-tpipe)
    - Default IOPCB Override values
      - LTERM and MODname
    - Timeout values (Socket and Execution)
    - Should reply include flag to indicate “asynchronous message waiting”
    - Should reply include flag to indicate “conversation ended”

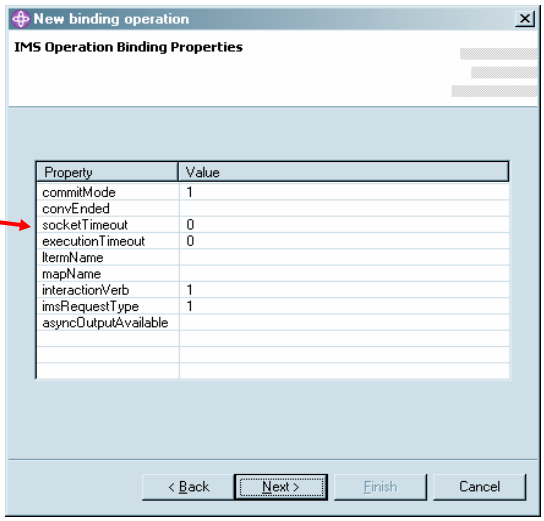
Connection Properties

Type of persistent socket for Commit Mode 0



Operation Binding Properties

New socket timeout value



Example

- COBOL Source

```
01  INPUT-MESSAGE.
    02  IN-LL              PICTURE S9(3) COMP.
    02  IN-ZZ              PICTURE S9(3) COMP.
    02  IN-TRANCODE        PICTURE X(9).
    02  IN-PERSON-NUMBER   PICTURE X(6).

01  OUTPUT-MESSAGE
    02  OUT-LL         PICTURE S9(3) COMP VALUE +0.
    02  OUT-ZZ         PICTURE S9(3) COMP VALUE +0.
    02  OUT-LASTNAME   PICTURE X(20) VALUE SPACES.
    02  OUT-FIRSTNAME  PICTURE X(20) VALUE SPACES.
```

```
public class OUTPUTMESSAGE extends
WSIFFormatPartImpl
```

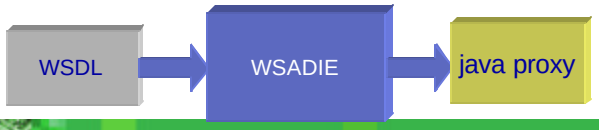
```
public class INPUTMESSAGE extends
WSIFFormatPartImpl
{
  .....
 public void setIn__ll(short in__ll)
 {
  this.basicSet("in__ll", 0, new Short(in__ll));
 }
  public void setIn__zz(short in__zz)
 {
  this.basicSet("in__zz", 0, new Short(in__zz));
 }
.....
```

- WSDL Message Definitions

```
.....
<element name="in_ll">
  <simpleType>
    <restriction base="short">
      <minInclusive value="-999"/>
      <maxInclusive value="999"/>
    </restriction>
  </simpleType>
</element>
<element name="in_zz">
  <simpleType>
    <restriction base="short">
      <minInclusive value="-999"/>
      <maxInclusive value="999"/>
    </restriction>
  </simpleType>
</element>
<element name="in_trancode">
  <annotation>
    <appinfo source="http://www.wsadie.c om/appinfo">
      <initialValue kind="SPACE"/>
    </appinfo>
  </annotation>
  <simpleType>
    <restriction base="string">
      <length value="9"/>
    </restriction>
  </simpleType>
  .....
```

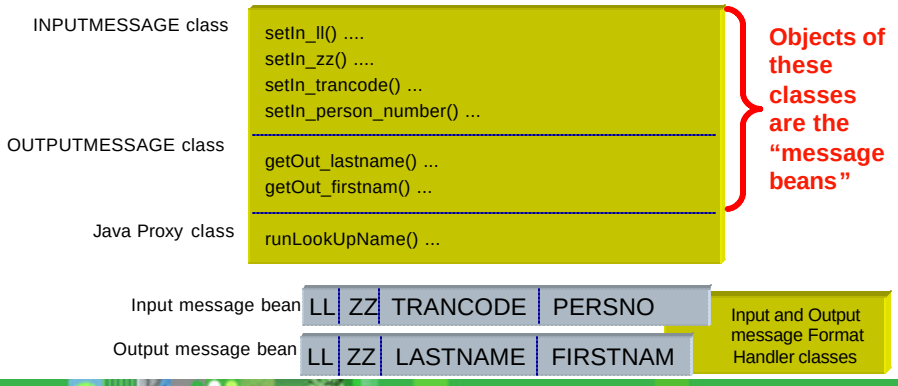
Building the Service …

- **WSADIE is then used (optional) to add to the WSDL any fields that the application will specify in addition to the actual input message itself**
  - **Client ID**
  - **Overrides of commit mode, LTERMname etc.**
- **To unit test the Enterprise Service, WSADIE is used to -**
  - **Build a Java Proxy**
    - **the class that contains the "run transaction" method**
  - **Write a java program that uses the methods of the INPUTMESAGE class to create an input message bean, puts values into the fields, calls the transaction (using the proxy) and displays the output message data**
  - **Run the unit test**



Generated Java Classes

- **The INPUTMESSAGE and OUTPUTMESSAGE classes contain methods (cf. sub-routines) for putting values into each field of the input message bean and retrieving the values of each field from the output bean**
- **The Proxy contains the method for running the transaction**
- **The Format Handlers convert between java beans and IMS messages**



Test Program

Create input and output message beans

Populate input bean

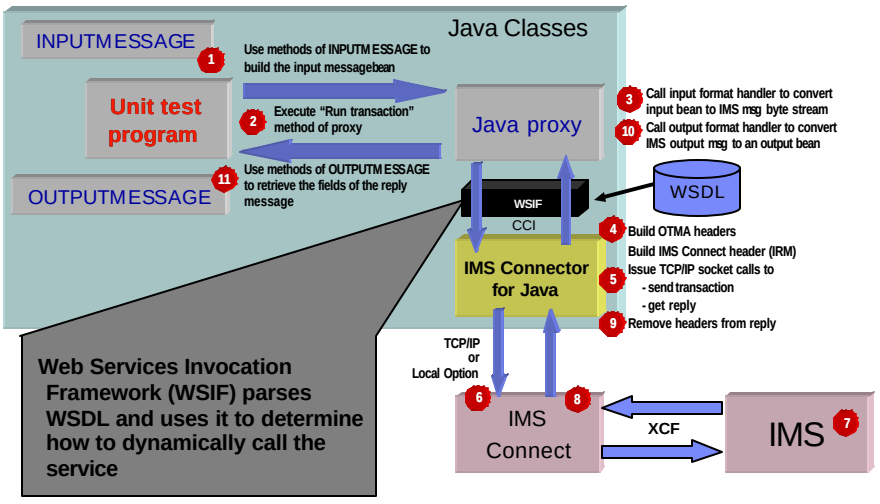
Run transaction

Handle output bean

```
.....
LookupNameProxy proxy = new LookupNameProxy();
INPUTMESSAGE input = new INPUTMESSAGE();
OUTPUTMESSAGE output = new OUTPUTMESSAGE();
/* build input message bean */
input.setIn_ll(18);
input.setIn_zz(0);
input.setIn_trancode("LOOKUP");
input.setIn_person_number("015771");
.....
/* execute transaction  */
output = proxy.runLookUpName(input);

/* display output in WSADIE console window */
..... output.getOut_firstname() ....
..... output.getOut_lastname() ...
```

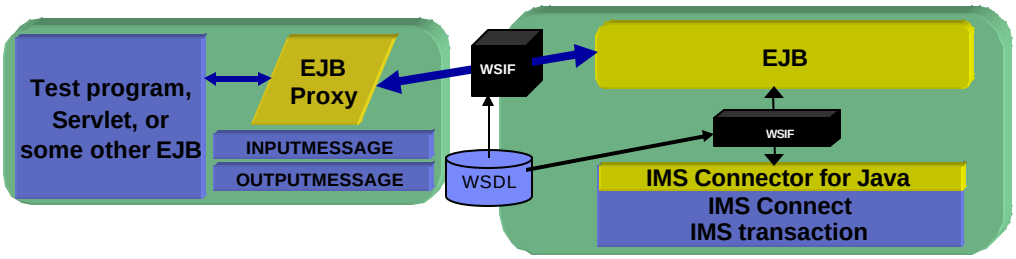
This program - with just a change to the name of the proxy class - will also be used to test EJB and/or SOAP (Web services) access to the transaction

How the Transaction is Invoked (in Test)

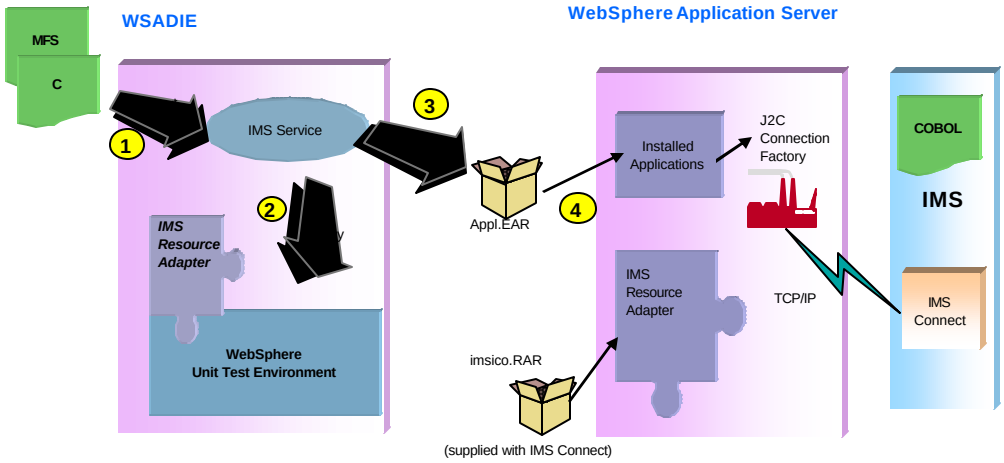


Deploying the Enterprise Service as an EJB

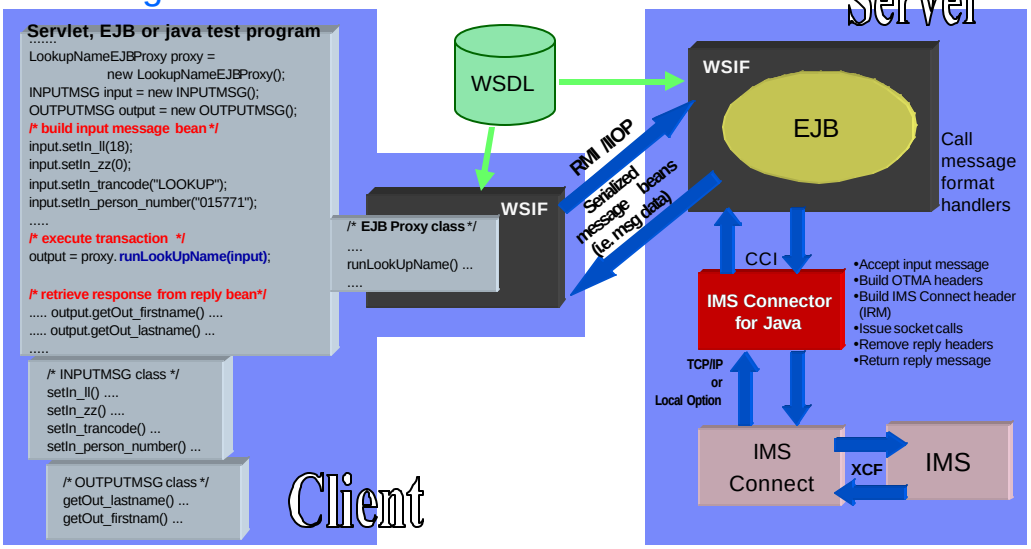
- **Again this is accomplished using WSADIE**
  - Select the appropriate WSDL file for the Enterprise Service
  - Select the option to generate the "deploy" code for EJB access
    - **WSADIE builds an Enterprise Archive (EAR) file containing the WSDL and the EJB, plus the other resource files needed to access the IMS transaction**
  - Create an "**EJB Proxy**"
  - Modify "unit test" to call the "run transaction" method of the EJB proxy



Develop & Deploy



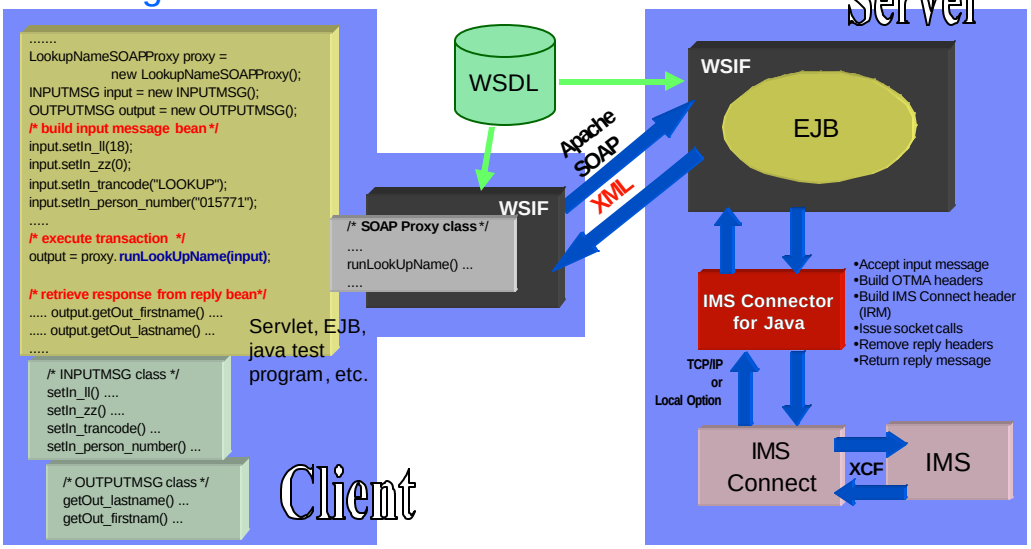
Invoking the Transaction via an EJB



Deploying the Enterprise Service as a Web Service

- **This is an extension of the EJB case**
  - Client service request is in SOAP XML format (instead of a java Remote Method Invocation (RMI))
- **Again this is accomplished using WSADIE**
  - Select the appropriate WSDL file for the Enterprise Service
  - Select the option to generate the "deploy" code for SOAP access
    - **WSADIE builds an Enterprise Archive (EAR) file containing the WSDL and the EJB, plus the other resource files needed to access the IMS transaction**
  - Create a "**SOAP Proxy**"
  - Modify "unit test" to call the "run transaction" method of the SOAP proxy

Invoking the Transaction as a Web Service



Summary

Summary – IMS Connector for Java

- **IMS TM can be accessed as an EIS within the J2EE environment**
  - **IMS Connector for Java (aka. IMS Resource Adapter) is the JCA connector for IMS**
  - **Highly secure (SSL, container or component sign-on)**
  - **Industrial strength**
  - **Scalable and flexible topology**
- **Each new release of IMS Connect and IMS Connector for Java provides enhancements to the already rich set of functions**
- **You can write applications to use the J2EE CCI for accessing EIS**
  - **IMS Connector for Java provides helper classes to simplify the development effort**
- **WSADIE enables rapid development of an IMS transaction as an Enterprise Service**
  - **Can be deployed as a simple EJB or as a Web (SOAP) Service**

Getting More Information

- **IMS, IMS Connect, IMS Connector for Java**
  - http://www.ibm.com/ims
    - Downloadable documentation
- **WebSphere Application Server**
  - http://www.ibm.com/software/webservers/appserv/was/
- **WebSphere Studio Application Developer Integration Edition**
  - http://www.ibm.com/software/awdtools/studiointegration/