E37

IMS Connector for Java: Developing J2EE Applications to Access IMS Transactions Using WebSphere Studio

James D. Polo

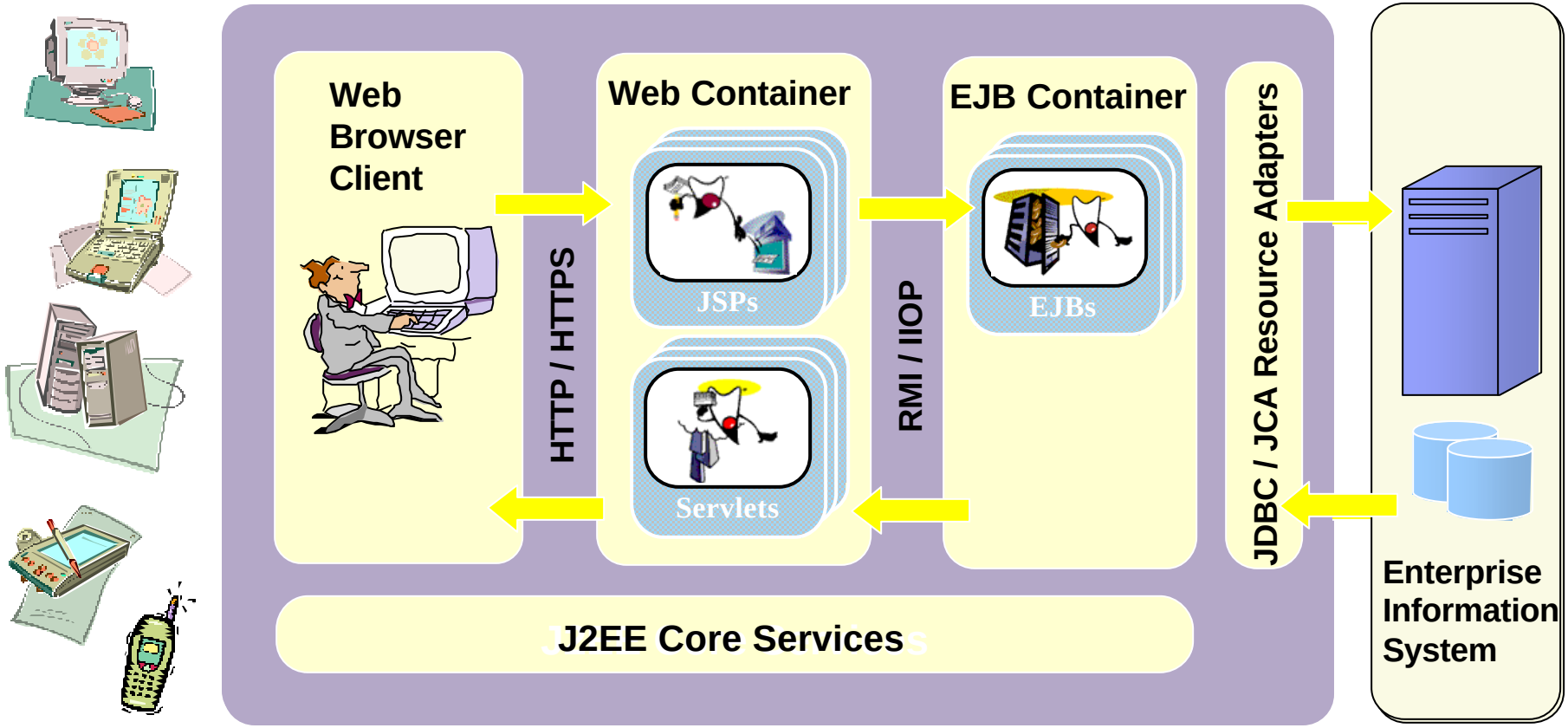
Las Vegas, NV **September 15 – September 18, 2003**

RETURN TO INDEX

Agenda

- J2EE
  - Server model
  - Connector architecture
- IMS Connector for Java
- Developing applications to access IMS transactions
  - Types of applications
  - Common Client Interface (CCI)
  - WebSphere Studio Application Developer Integration Edition (WSADIE)
  - Demo
- Summary

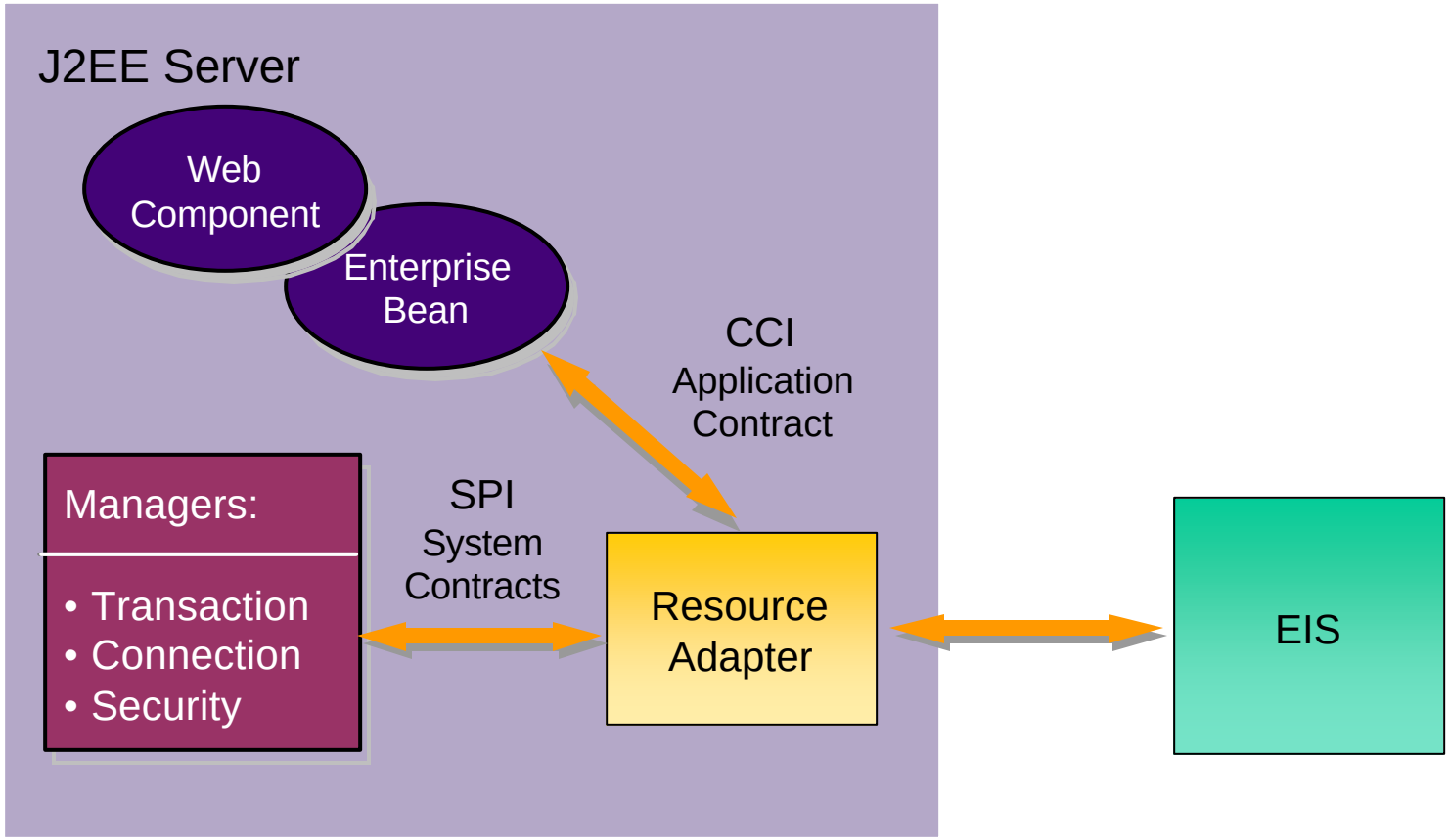
J2EE Server Model



J2EE Connector Architecture

- A standard architecture for Enterprise Information System (EIS) integration
- Connects J2EE platform to heterogeneous EISs
- Defines a common set of interfaces (SPI and CCI) that standardizes interactions between
  - Resource Adapters
  - Application Servers
  - Application Components
- Simplifies application component development
- No longer need to customize a connector for each application server, and vice versa

J2EE Connector Architecture



Agenda

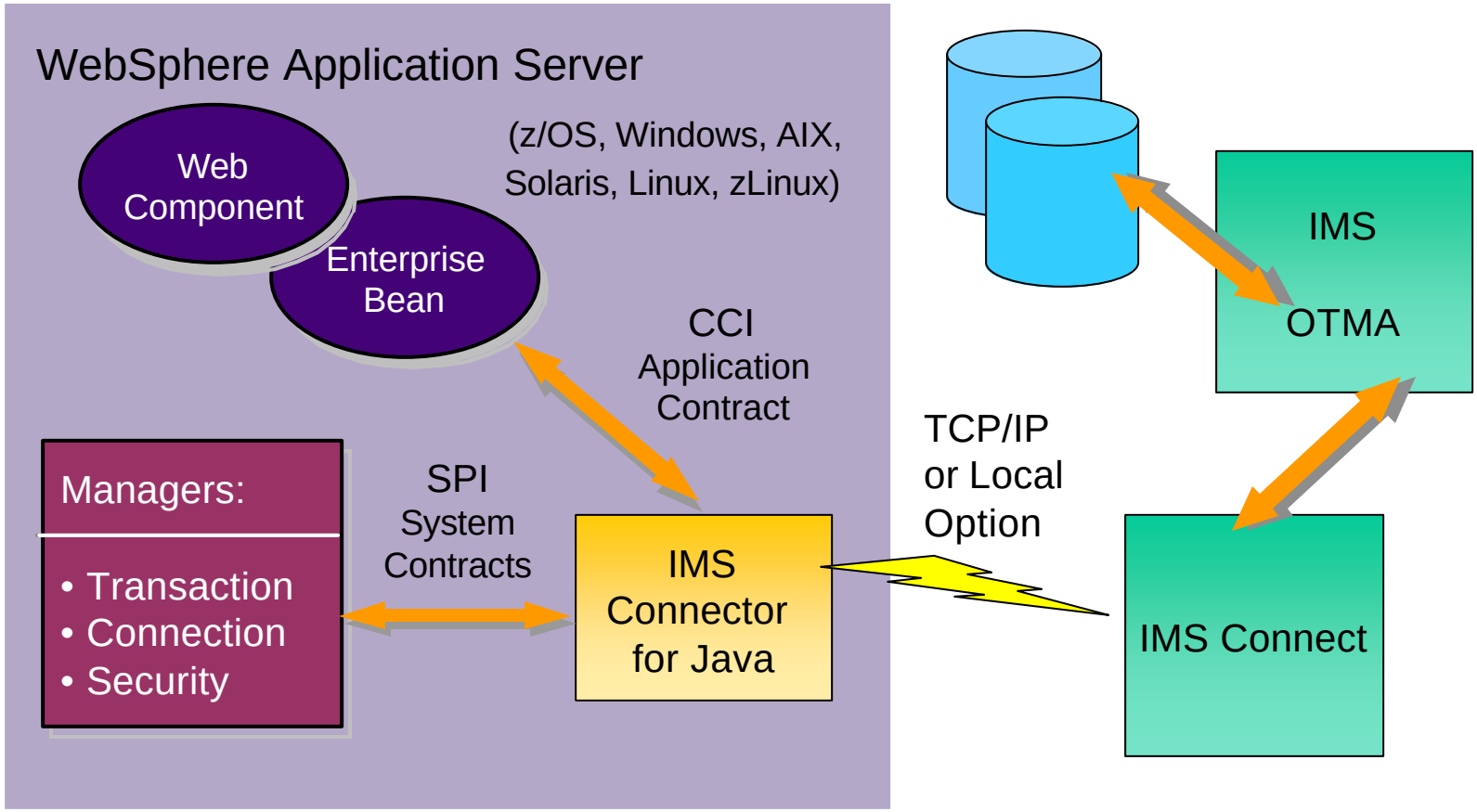
- J2EE
  - Server model
  - Connector architecture
- IMS Connector for Java
- Developing applications to access IMS transactions
  - Types of applications
  - Common Client Interface (CCI)
  - WebSphere Studio Application Developer Integration Edition (WSADIE)
  - Demo
- Summary

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

Helps IMS Users make the transition to ebusiness easier

J2EE Connector Architecture



IMS Connector for Java Features

- J2EE Connector Architecture resource adapter
- Rapid application development with WebSphere tooling
- Highly scalable and flexible topology
- Runs on both distributed and z/OS platforms
- TCP/IP and Local Option connection
- IMS conversational and non-conversational transactions
- Send-then-commit (Commit Mode 1)
- Local z/OS RRS 2-phase commit
- Container and Component Managed Sign-on

- Distributed XA 2-phase commit
- Message Format Service (MFS)
- Secure Socket Layer (SSL)
- Commit-then-send (Commit Mode 0)
- Retrieve asynchronous output messages
- Execution timeout

New Features Highlights

- 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
- MFS
  - Enables you to leverage investment in existing MFS-based IMS transactions by providing you with the ability to publish your applications as enterprise services
- SSL
  - Provides reliable, secure communication between IMS Resource Adapter and IMS Connect

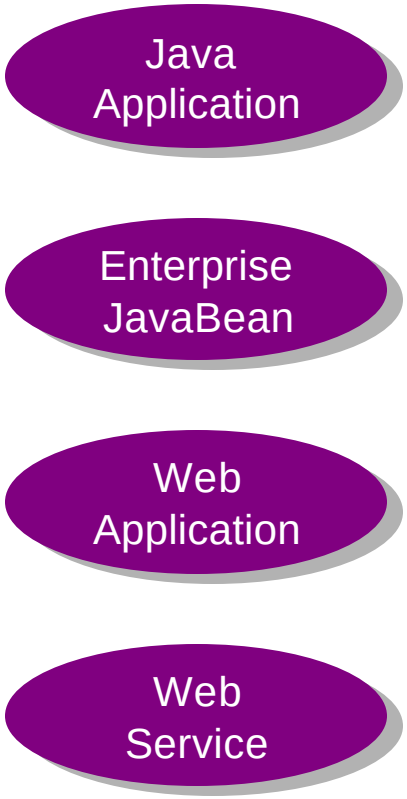
New Features Highlights

- Commit-then-send (Commit Mode 0)
  - Allows clients to run IMS transaction with recoverable output message
- Retrieve Asynchronous output message (Resume Tpipe)
  - Retrieve asynchronous output messages from IMS OTMA Asynchronous Hold Queue
- Execution Timeout
  - Control the maximum amount of time allowed for IMS Connect to send a message to IMS and receive a response from IMS on a per interaction basis

Agenda

- J2EE
  - Server model
  - Connector architecture
- IMS Connector for Java
- Developing applications to access IMS transactions
  - Types of applications
  - Common Client Interface (CCI)
  - WebSphere Studio Application Developer Integration Edition (WSADIE)
  - Demo
- Summary

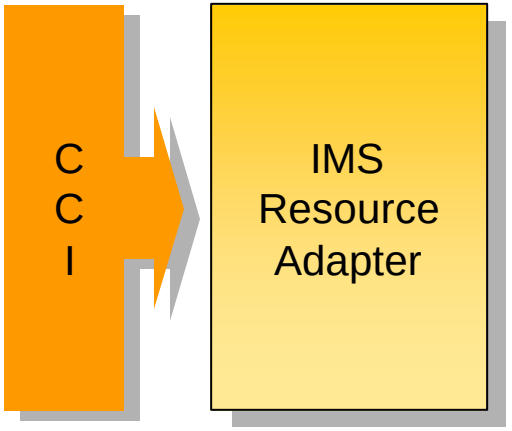
Developing Applications



- 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

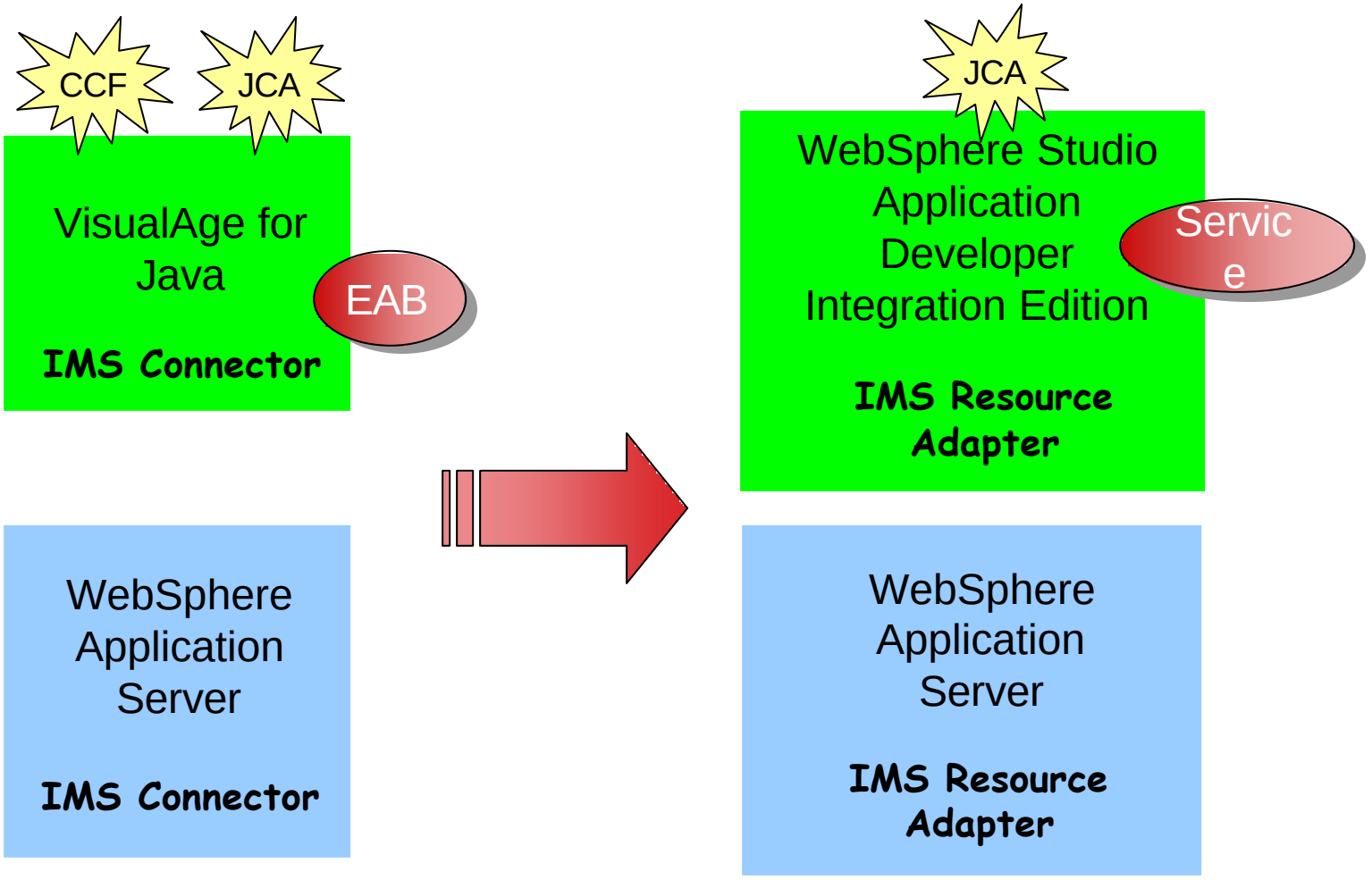


(The generated code uses the CCI to interact with the Resource Adapter)

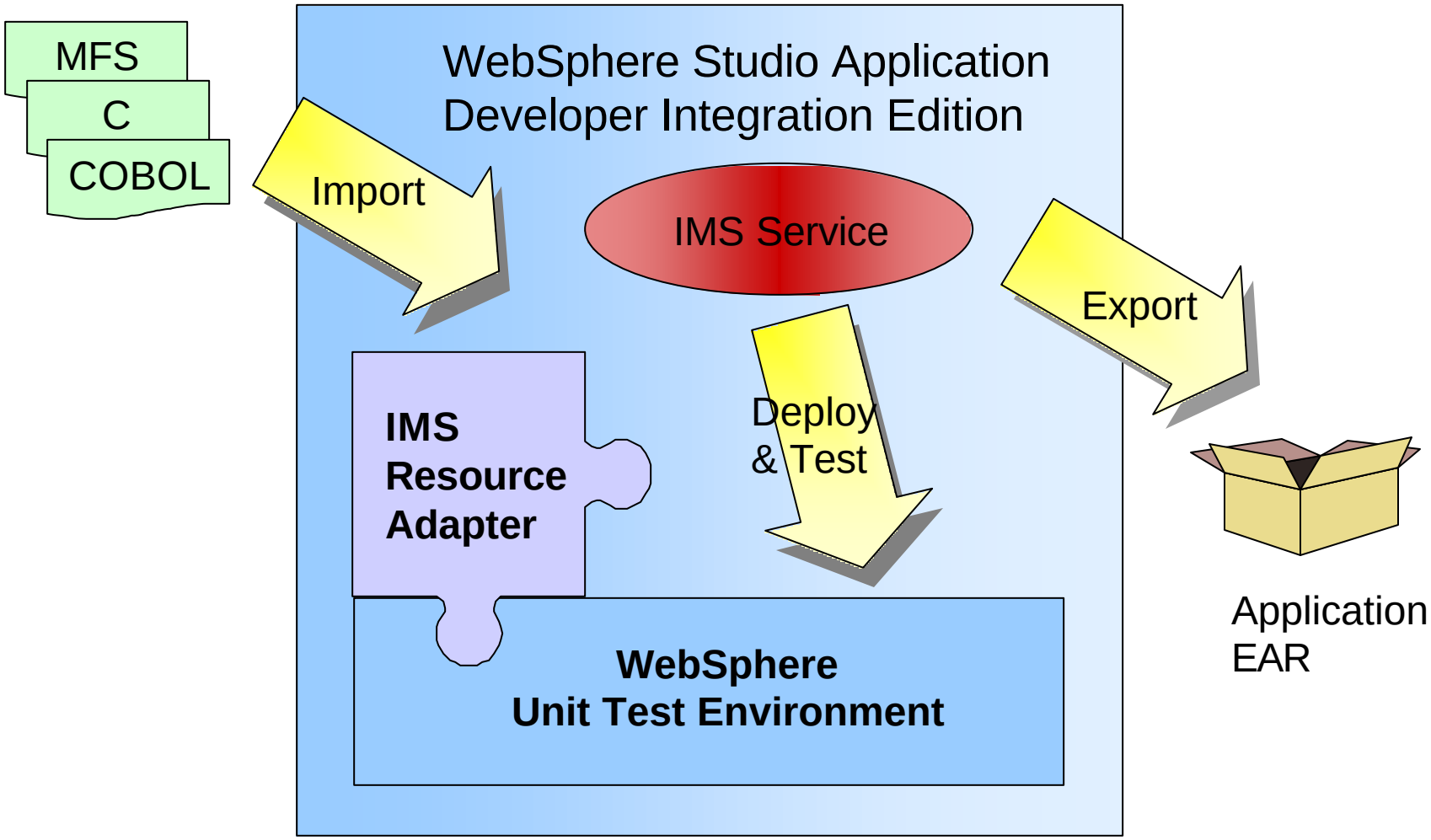
Using the Common Client Interface (CCI)

```
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
```

WebSphere Studio Tooling Evolution

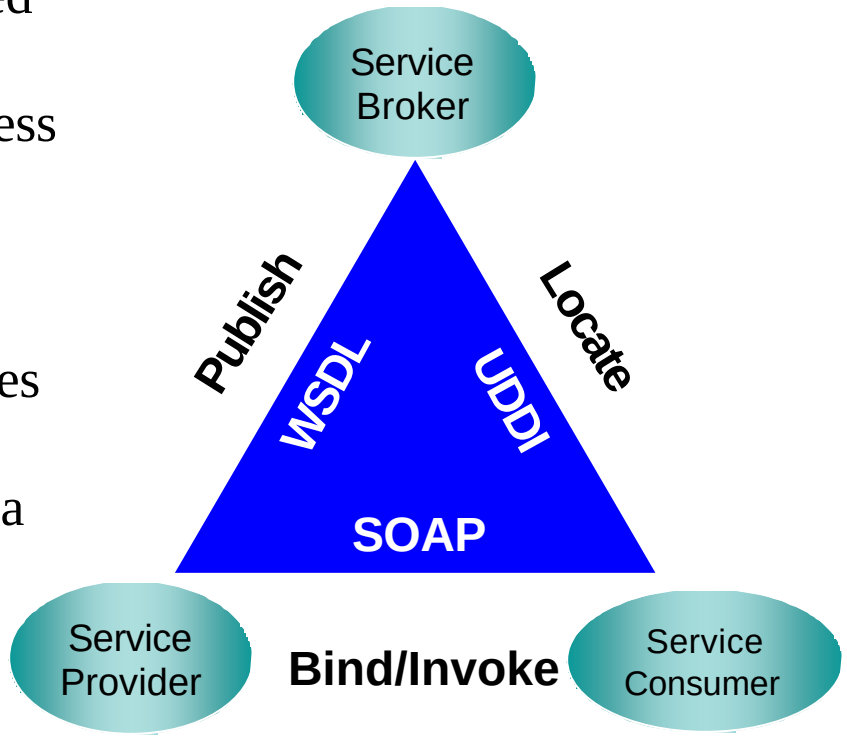


WSAD IE Development



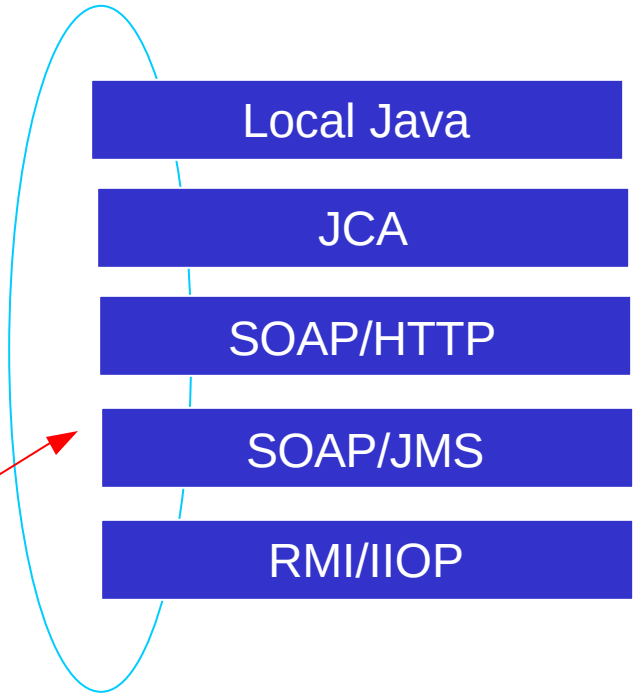
What are Web Services ?

- Self-contained, self-describing modular applications
- Can be published, located, and invoked across the Web
- Are reusable building blocks of business functionality
- Web services uses simple, standard interfaces for
  - Publishing and discovering services
  - Communicating between service provider and service consumer in a platform-independent way



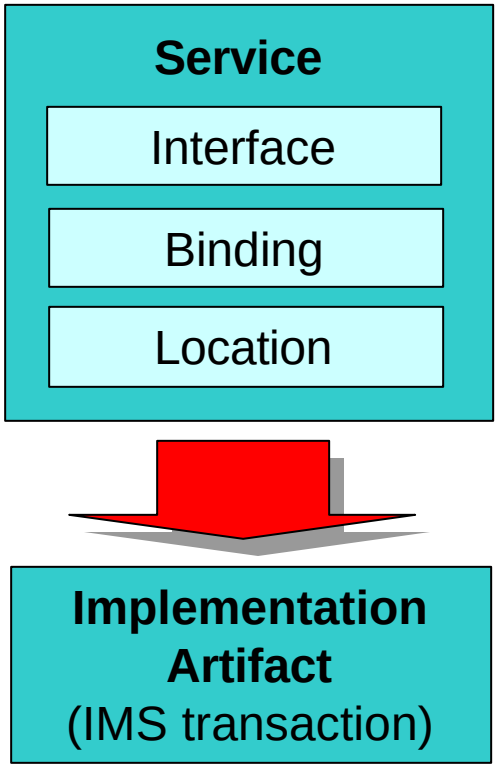
What is an Enterprise Service ?

- WSAD IE is based on the concept of an **enterprise service** (also referred to as service)
- Like Web services, services:
  - Transform software into components that are invokable remotely
- Unlike Web services, services:
  - Do not restrict bindings to Web protocols (such as SOAP over HTTP)
  - Support many transport and data encapsulation layers (SOAP over JMS, EJB, Java, etc…)
- **A Web service is just a type of service**



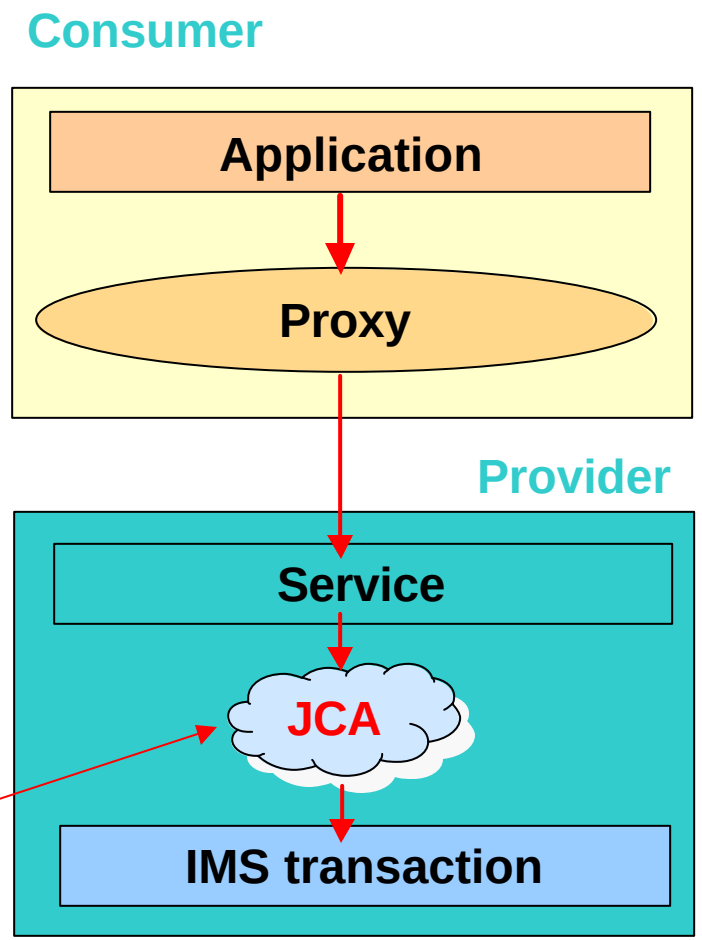
The Service Provider Builds the Service

- A service consists of:
  - An implementation artifact (e.g., a Java class, EJB, IMS transaction, etc.)
  - 3 WSDL files that 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 implemented
    - **Implementation service**
      - Location of the service
- To Generate the service (WSDL)
  - Import C, COBOL, MFS description of transaction input and output messages
  - Provide connection, interaction properties



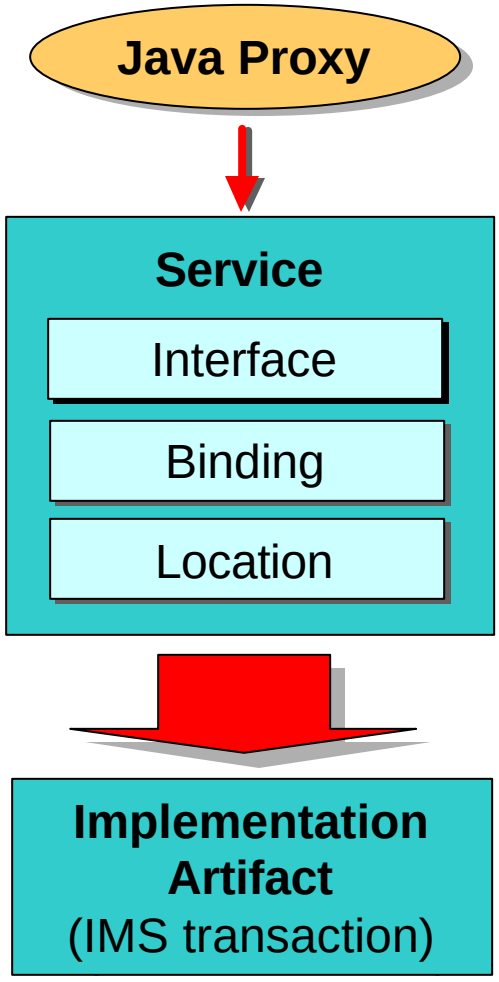
The Service Consumer Uses the Service

- Generate a **proxy** for the consuming application to use
  - Represent the service that is deployed to the server
  - Handle the sending and receiving of data to the service
  - Expose only the methods of the service the application needs
- The proxy uses Web Services Invocation Framework (**WSIF**)
  - WSIF parses WSDL and dynamically calls the service



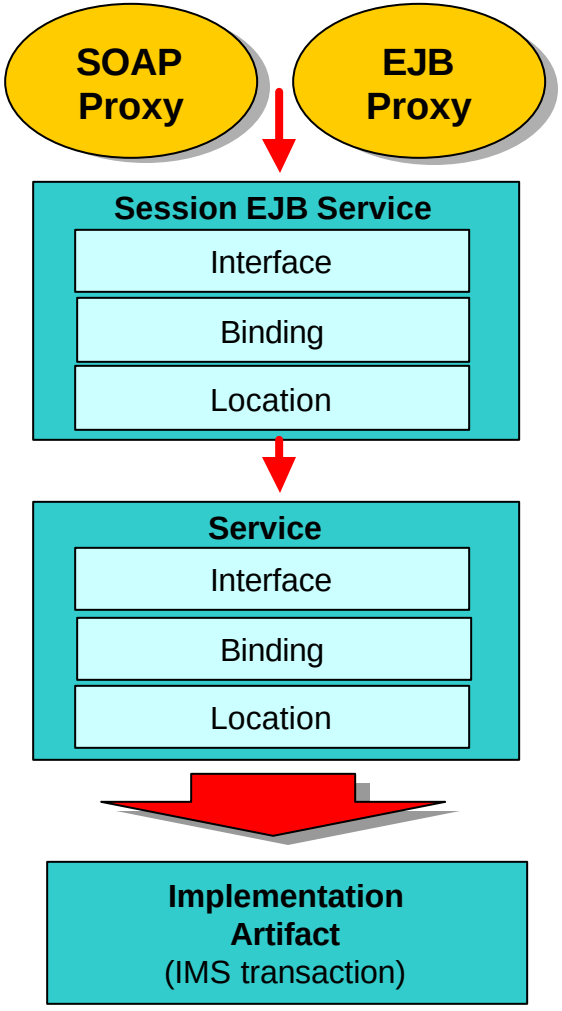
Types of Service Proxies

- **Java proxy**
  - Use tooling to generate a Java proxy for the enterprise service
  - Test enterprise service in “two-tier” environment
    - Invoke Java proxy from a simple Java class
    - No application server (non-managed)
    - No J2EE Quality of Service (QoS)
      - For example, no connection pooling



Types of Service Proxies

- **EJB or SOAP proxy**
  - Use to access a deployed service running in WebSphere Application server
  - Implemented by a **stateless session bean** (EJB)
  - EJB synchronously handles the client requests to and responses from the local service
  - J2EE component (EJB) means managed environment (e.g., connection pooling)
  - Described by inbound binding WSDL files
    - How the service is provided to consumers (clients) – SOAP or EJB
  - Wizards generate proxy and deployment descriptor for EJB



Application Development Demo

Agenda

- J2EE
  - Server model
  - Connector architecture
- IMS Connector for Java
- Developing applications to access IMS transactions
  - Types of applications
  - Common Client Interface (CCI)
  - WebSphere Studio Application Developer Integration Edition (WSADIE)
  - Demo
- Summary

Summary – IMS Connector for Java

- Based on J2EE Connector Architecture open standard
- Supports rapid development of applications that access IMS transactions
  - WebSphere Studio Application Developer Integration Edition
- Offers a highly scalable and flexible topology
  - IMS Connect and WebSphere Application Server (z/OS and distributed platforms)
- Supports development of secure, industrial strength e-business applications
  - For example: SSL, two-phase commit

Finding IMS Connector for Java

- IMS Connector for Java development
  - Latest versions are included in WebSphere Studio Application Developer Integration Edition (WSADIE)
- IMS Connector for Java runtime
  - A component of the IMS Connect product
  - For distributed platforms
    - Downloadable from IMS Web site (http://www.ibm.com/ims)
  - For z/OS platform
    - SMPE installable
- Also known as the *IMS Resource Adapter* or the *WebSphere Adapter for IMS*

Getting More Information

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