IBM Information
>>> On Demand

IBM

IMS in the World of SOA

Alan Cooper IBM UK
alan_cooper@uk.ibm.com

TAKE BACK CONTROL

Important Disclaimer

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY.

WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED.

IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE.

IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION.

NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, OR SHALL HAVE THE EFFECT OF:

- CREATING ANY WARRANTY OR REPRESENTATION FROM IBM (OR ITS AFFILIATES OR ITS OR THEIR SUPPLIERS AND/OR LICENSORS); OR
- ALTERING THE TERMS AND CONDITIONS OF THE APPLICABLE LICENSE AGREEMENT GOVERNING THE USE OF IBM SOFTWARE.

Agenda

- **Quick review of SOA and the role of Legacy applications**
- **Overview of the tools used to develop IMS SOA solutions**
- **Overview of SOA with IMS Transactions, and the IMS TM options supported**
- **Using the tools to generate SOA solutions with IMS TM**
  - Rational Application Developer with WebSphere Application Server
  - Rational Developer for System z with IMS SOAP Gateway
  - WebSphere Integration Developer with WebSphere Process Server
- **Callout from IMS to external services**
- **Writing java services that directly access IMS DBs**
  - With IMS java
  - With WebSphere Classic Federation Server for z/OS
  - With IMS 10 Service Data Object support

Service Oriented Architecture Introduction

- **The Primary Goal**
  - **Align the business world with the world of information technology in a way that makes both more effective**

- **SOA is not just about technology**
  - IBM views SOA as a holistic relationship between the business and the IT organization
- **SOA starts from the premise that all businesses have a *business design…***
  - … which describes how the business works – processes, structures, finances, goals and objectives, influencing factors, rules and policies
- **A written business design becomes an essential tool in communicating requirements between the business and the IT organization**
- **WebSphere and Rational tooling is available that enables a business model to be created (and maintained), and an IT model to be derived from it**

SOA and Services Terminology

… a service?

A **repeatable business task** – e.g., check customer credit; open new account

… service orientation?

A way of integrating your **business as linked services**

… service oriented architecture (SOA)?

An IT **architectural style** that supports service orientation

… a composite application?

A set of **related & integrated** services that support a business process built on an SOA

Ways to Implement a Service

- **A service can be represented by an MQ queue, and implemented by a message driven java bean, an IMS transaction, etc.**
- **A service can be implemented by a java EJB**
- **But an open, standard method is to implement the service as a Web Service**
  - Independent of hardware or software platform
  - A standard way of both defining and calling the service
  - Especially relevant when there already exists a range of technologies or where it is appropriate to publish the service for use by third parties
- **A service can also be implemented as a component within a Service Component Architecture (SCA) solution**

Service Component Architecture (SCA)

- **SOA is not just about services, but about Processes composed of services**

- **Service Component Architecture simplifies the building and implementing of Services and Processes**
  - Hides the technology from the programmer, allowing him to concentrate on the business logic, and not technology
  - Allows a service component to be written in Business Process Execution Language (BPEL), or Java, (or other languages)
  - With java, SCA enables Plain Old Java Objects (POJOs) or EJBs or Web Services to be used to implement services
  - Services can call other services using a standard SCA API

- **SCA Services and Processes are assembled using WebSphere Integration Developer and are deployed in WebSphere Process Server**

The Role of Legacy in SOA

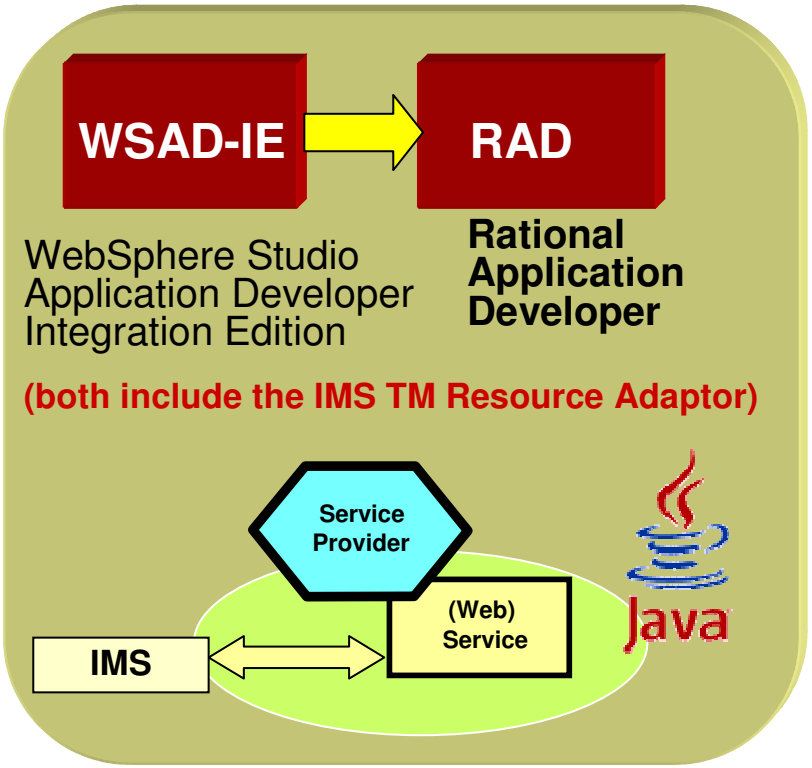
- **In most cases, much of the business design will already be implemented with existing IT application systems**
  - These applications were probably the result of formal systems analysis …
  - … though perhaps based on a blinkered view of the overall business, …
  - … without consideration for reuse and integration across lines of business

- **Nevertheless, there is a good chance that you already have applications that implement the services in the business design**
  - **Legacy is GOOD!**
- **This is especially true of IMS transactions - they were written to satisfy specific business functions, and rarely contain any presentation logic**
- **Modelling the business design will either validate the existing applications or indicate where change is needed**
  - Modifications to existing code, or
  - New application code

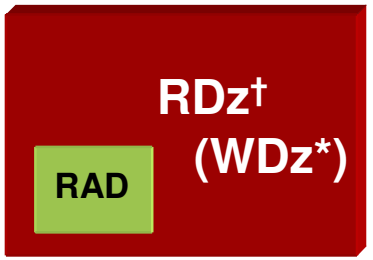
> *"The SOA Foundation architecture embraces legacy as a tremendously valuable asset and deliberately avoids requiring that you re-engineer that entire legacy into a new generation of technology or language"*

IBM Tools for Building IMS-based Services

Toolkits that generate Web Services, EJBs, JSPs, etc. for IMS



WebSphere Integration Developer
Used to assemble Service Component Architecture (SCA) services to run in WebSphere Process Server



Rational Developer for System z
COBOL and PL/1 workbench and an integrated set of tools that supports end-to-end, model-based application development

† two free licences for WDz or RDz are made available to all IMS 10 customers.

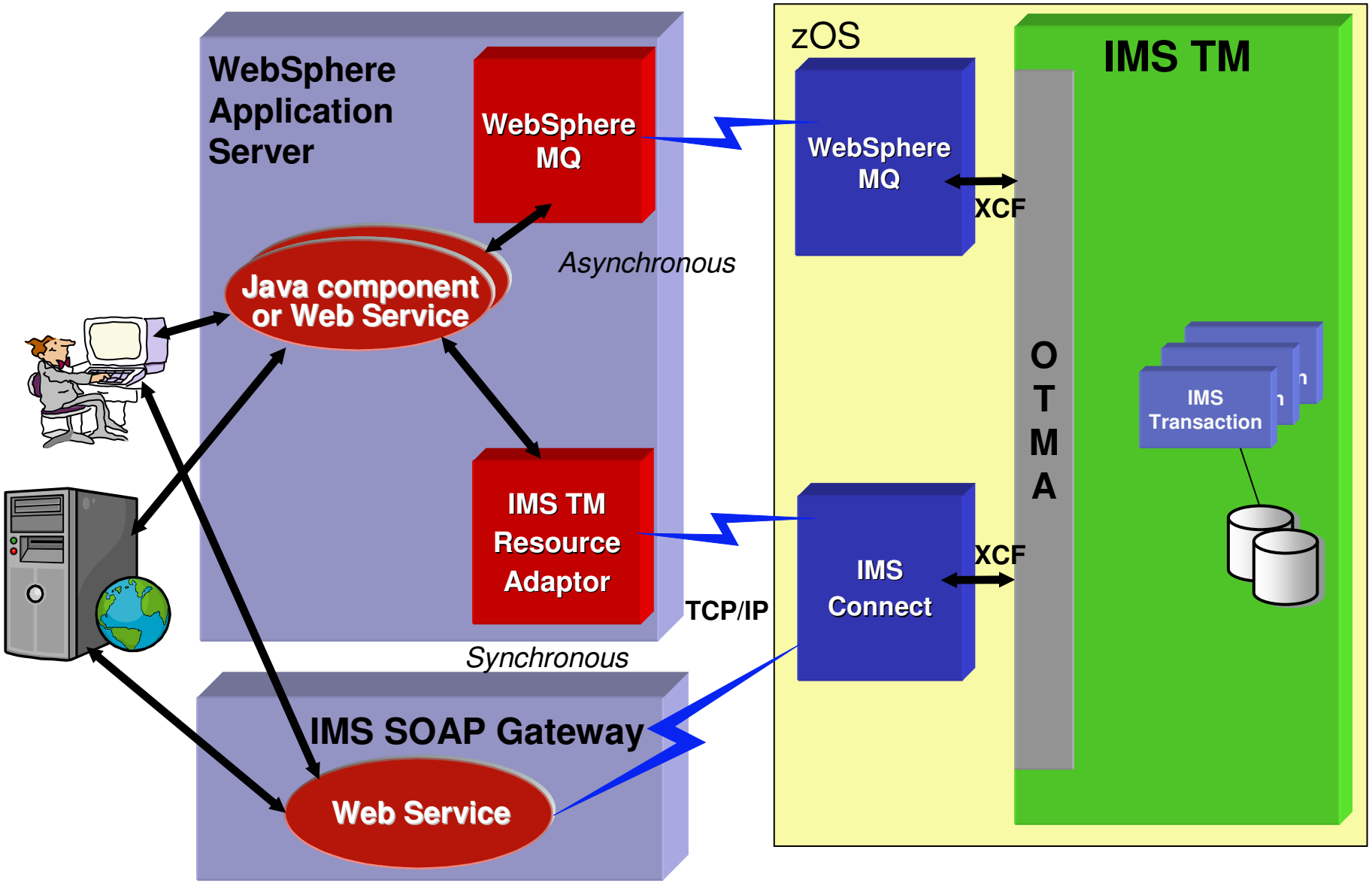
* The previous version of RDz was called WebSphere Developer for System z (WDz)

Using Tooling with IMS Transactions

- Where each service is represented by an MQ queue, an IMS Transaction is the service, and is accessed via the IMS MQ Bridge
- In other SOA solutions, the service is represented by a java component that calls the IMS transaction
  - RAD is used to create an EJB or Web Service that runs in WAS and calls an IMS transaction
  - RDz (or WDz) is used to create a Web Service that runs in the IMS SOAP Gateway and calls an IMS transaction
  - WID is used to create an SCA service that runs in WebSphere Process Server and either –
    - calls an IMS transaction

    or
    - invokes a Web Service (e.g. built with RAD) that calls an IMS transaction
- These tools all have a common "look and feel" and create the IMS service from the IMS transaction message structures used by the application source code

The IMS SOA Implementations

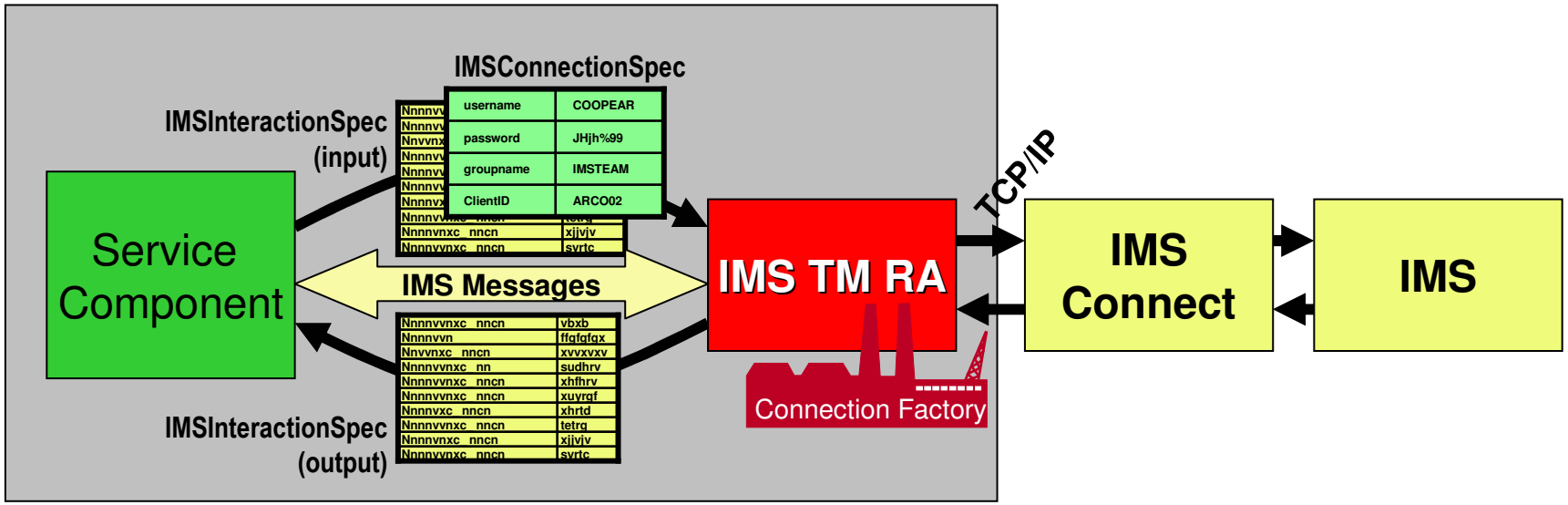


Support for IMS TM Options

- **IMS TM allows a range of transaction functions – but all these are supported by services built with RAD that invoke IMS TM**
  - Single or Multi-segment messages
  - Conversational transactions
  - Input-Only transactions
  - Output-Only requests
    - Asynchronous retrieval of undelivered transaction replies or ALTPCB messages
  - Input of LTERMname and/or Map Name (MODname) for IOPCB
  - Output of Map Name used on IOPCB ISRT

Calling an IMS Transaction from WAS

- **Most of the java code needed to call an IMS transaction via IMS Connect is in the IMS TM Resource Adapter**
- **The Service component invokes an IMS transaction by executing an IMS TM Resource Adapter method, and passes it three java objects –**
  - The input message bean and two sets of properties (parameter blocks)
- **The IMS TM RA extracts the message data from the message bean, uses the properties to build the OTMA and IMS Connect headers, and handles the TCP interaction with IMS Connect**

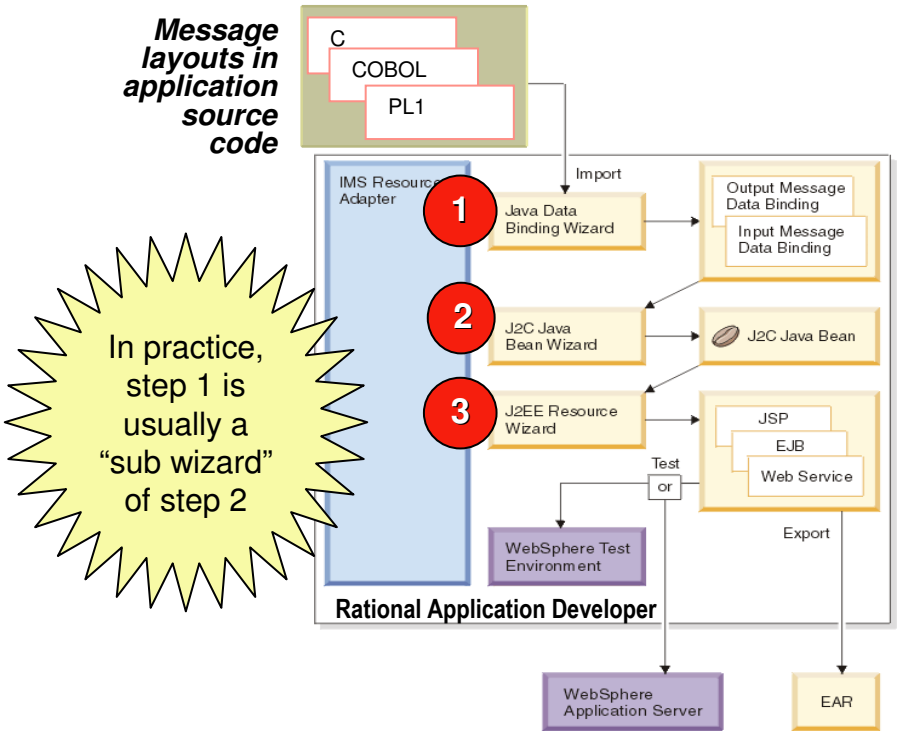


Transaction Properties

- **Properties passed to the IMS TM RA can either be hard coded in the service component, or can be *exposed* and provided by the caller of the service**
- **These properties include –**
  - Optional UserID and password
  - Type of input (e.g. send & receive, send only, asynchronous retrieval)
  - Timeout values
  - Reply message options - Commit Mode (Commit-then-Send or Send-then-Commit) and Sync_Level (NONE or CONFIRM)
  - Rerouting controls for undelivered replies
  - Asynchronous retrieval specifications
  - LTERMname to be passed to IMS application program
- **Properties that can be returned to the service component with the reply message bean include –**
  - Map name associated with the IMS reply (set by IMS application program)
  - End-of-conversation indicator

Generating Services with RAD for IMS TM Access via WAS

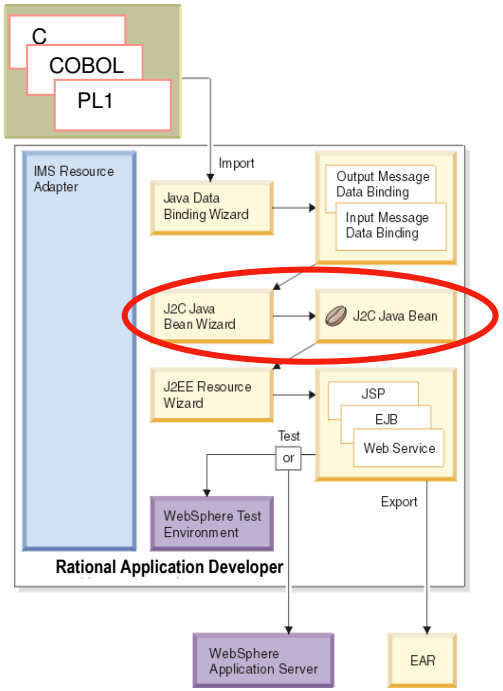
This is for the default Synchronous solution

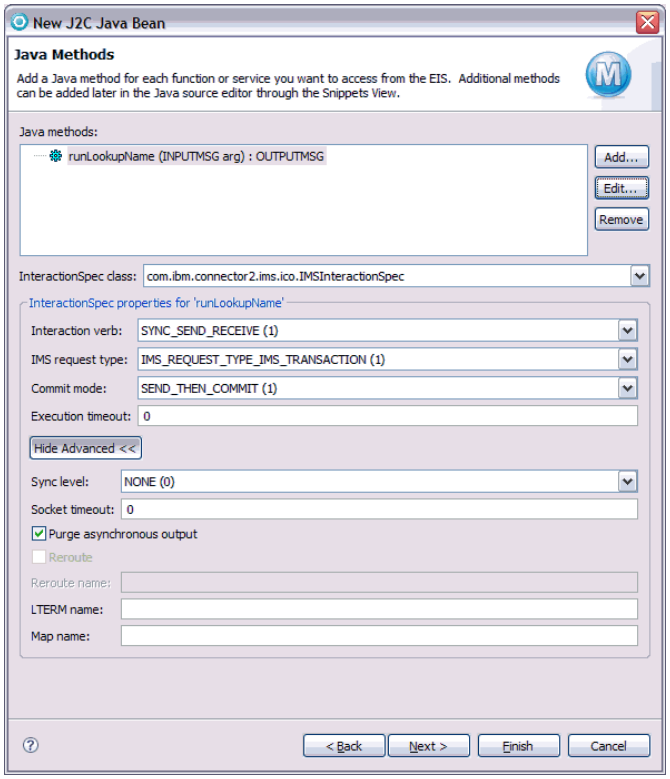


- **Step 1**
  - Import IMS application source code of message structures
  - Generate **Data Bindings**
    - Classes whose objects (beans) are the message segments
    - Get and Set Methods for all the fields in the message segments
- **Step 2**
  - Generate **J2C Bean**
    - The java code that receives an input message bean (built by the caller), calls the IMS transaction via the IMS TM Resource Adaptor, receives the reply message, and returns an output message bean to the caller
- **Step 3**
  - Deploy J2C bean as a service component (JSP, EJB or Web Service)

More about Step 2

- **In the process of building the J2C Bean, the RAD wizard will request the various properties, for example**
  - Type of transaction
    - send & receive
    - input only
    - retrieve queued message
  - Commit Mode – 0 or 1
  - Timeout values
  - LTERM and MODNAME values for IOPCB
  - Etc.

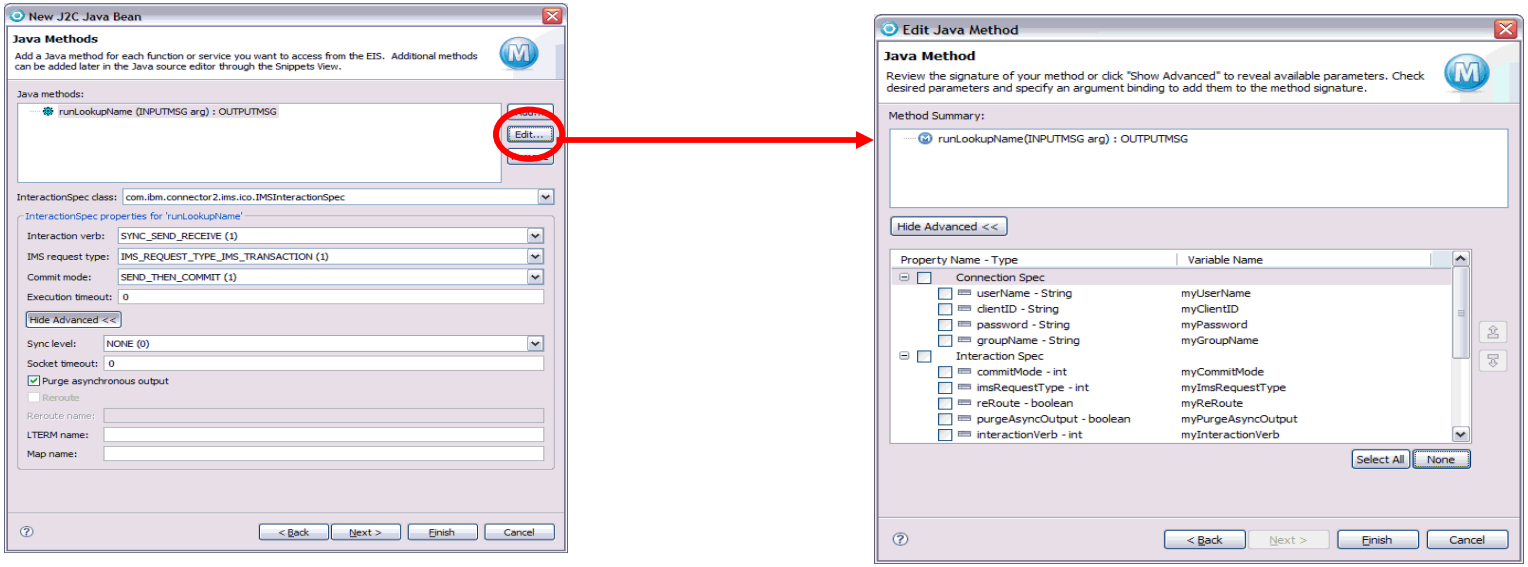




Requires team work by IMS application specialist and Service developer

More about Step 2 …

- **Alternatively, you can expose selected input properties by "ticking the boxes" available by clicking the "EDIT method" button**



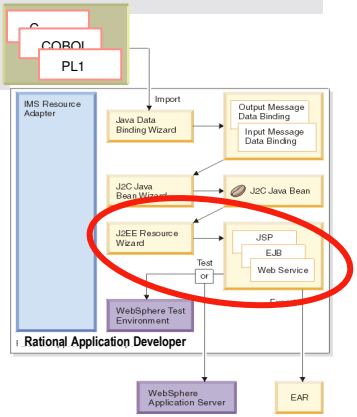
- **Exposing output properties requires a little java coding – creating a java class that contains the reply message bean and the required output properties**
  - RAD generates extra J2C code from programmer specified comments
  - A "5 minute task"

Supporting Other IMS Options with RAD

- **Messages with repeating groups**
  - Handled automatically by RAD's data bindings wizard
    - Each repeating group is a separate object within the message bean
- **Multi-segment Messages**
  - Use RAD data bindings wizard to generate a class for each message segment
  - Code a class that contains the multiple segment objects
    - Specified as input or output when generating the J2C bean
- **Conversations**
  - Instead of the J2C bean just having a single method to run the transaction, it should have additional methods generated. For example –
    - Run initial transaction of conversation
    - Run intermediate transaction of conversation
    - End conversation
  - It should also expose the output property, "Conversation Ended indicator"
  - Prior to IMS 10, a single J2C bean handles the whole conversation and ties up a TCP/IP socket. IMS 10 introduces more flexibility by allowing multiple J2C beans to be involved, using a conversation token supplied by IMS with first reply message of conversation

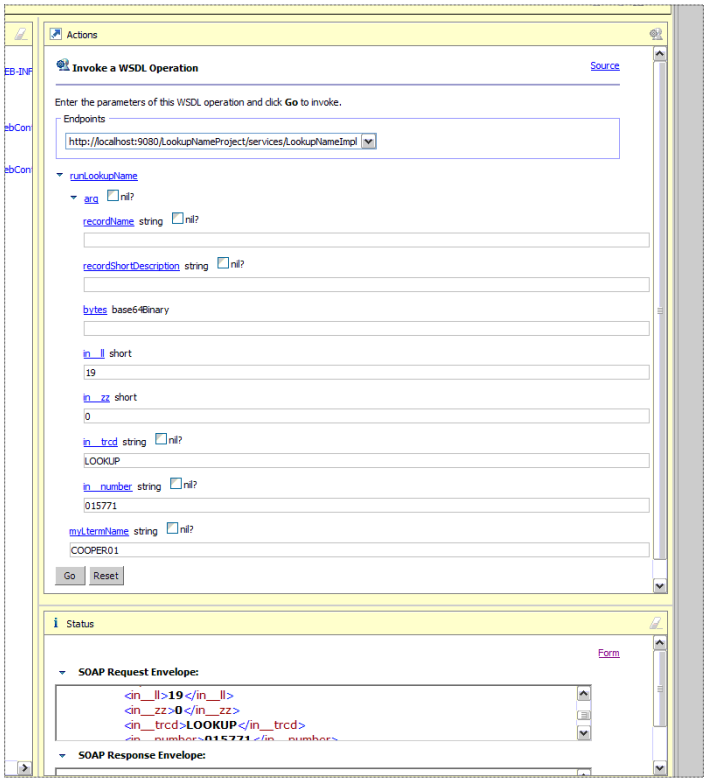
Validating the Service with RAD

- RAD includes an integrated WebSphere Application Server
- The third RAD wizard mentioned earlier allows the J2C bean to be deployed into the integrated WAS as a JSP, an EJB or a Web Service
- RAD generates a GUI to test the Service



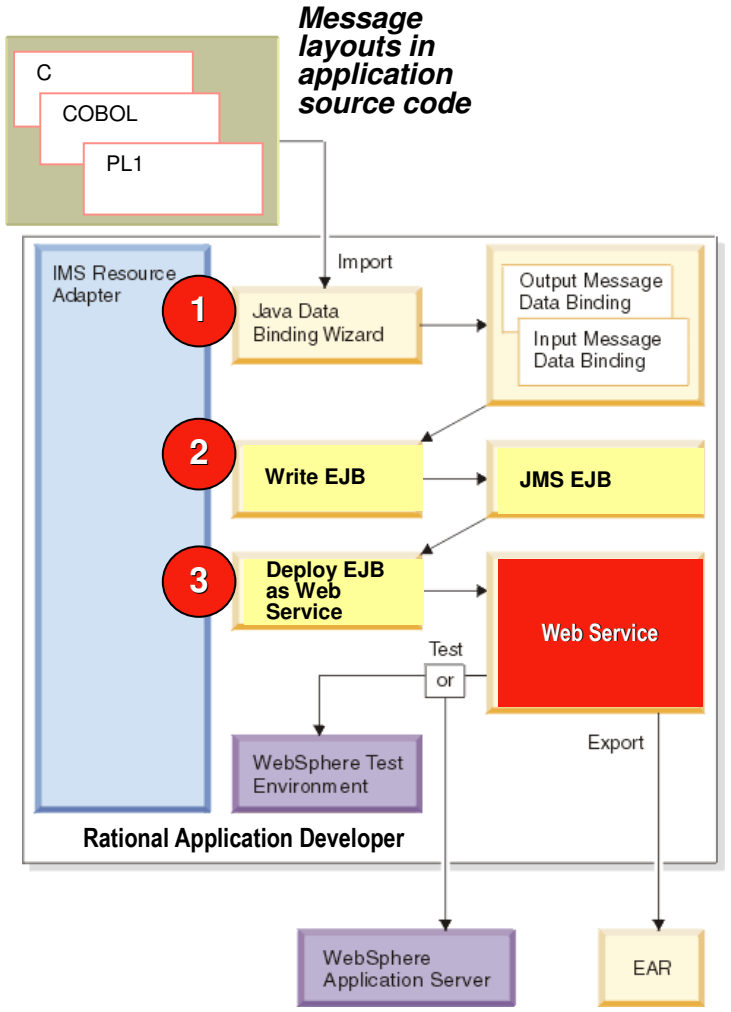
JSP Test

Web Service Test



Generating Web Service for Asynchronous IMS Access

Calling an IMS transaction from the Java component that implements the service is effectively like “calling a sub-routine”. Logically, this should be done synchronously via IMS Connect. However, an asynchronous solution via MQ is available, if so desired.

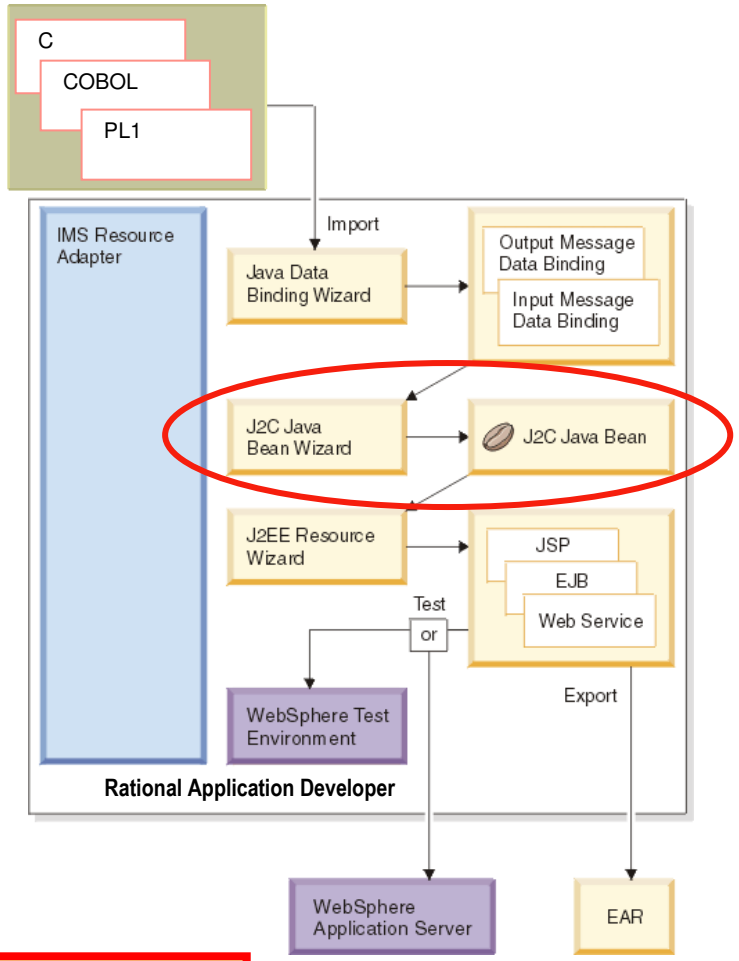


Uses a combination of RAD wizards and Java coding

- **Step 1**
  - Import IMS application source of message structures
  - Use wizard to generate Data Bindings
- **Step 2**
  - Code an EJB which uses Java Messaging Service (JMS) to call IMS transaction
- **Step 3 (optional)**
  - Use wizard to deploy EJB as a Web Service

More about Step 2

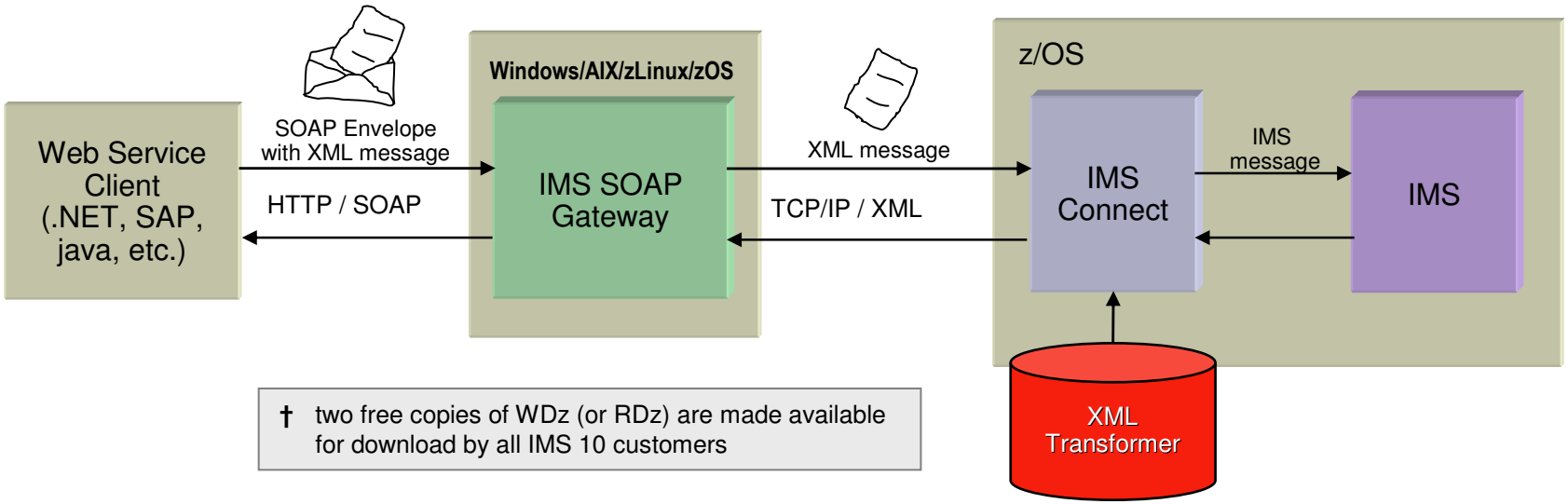
- **To use JMS to call the IMS transaction over MQ requires writing the code that:**
  - builds the MQ input message from the input message bean using the data bindings created by RAD
    - CAN include the optional input header (MQIIH) for the IMS MQ Bridge (to specify LTERM or Map Name for IOPCB)
  - uses JMS to invoke the transaction
  - uses the output data bindings to build the reply message bean from the MQ reply message
    - Can exploit data provided by IMS in the MQIIH if one was included with input (e.g. Map Name)



Requires team work by IMS application specialist and Service developer

IMS SOAP Gateway (for Synchronous IMS TM Access)

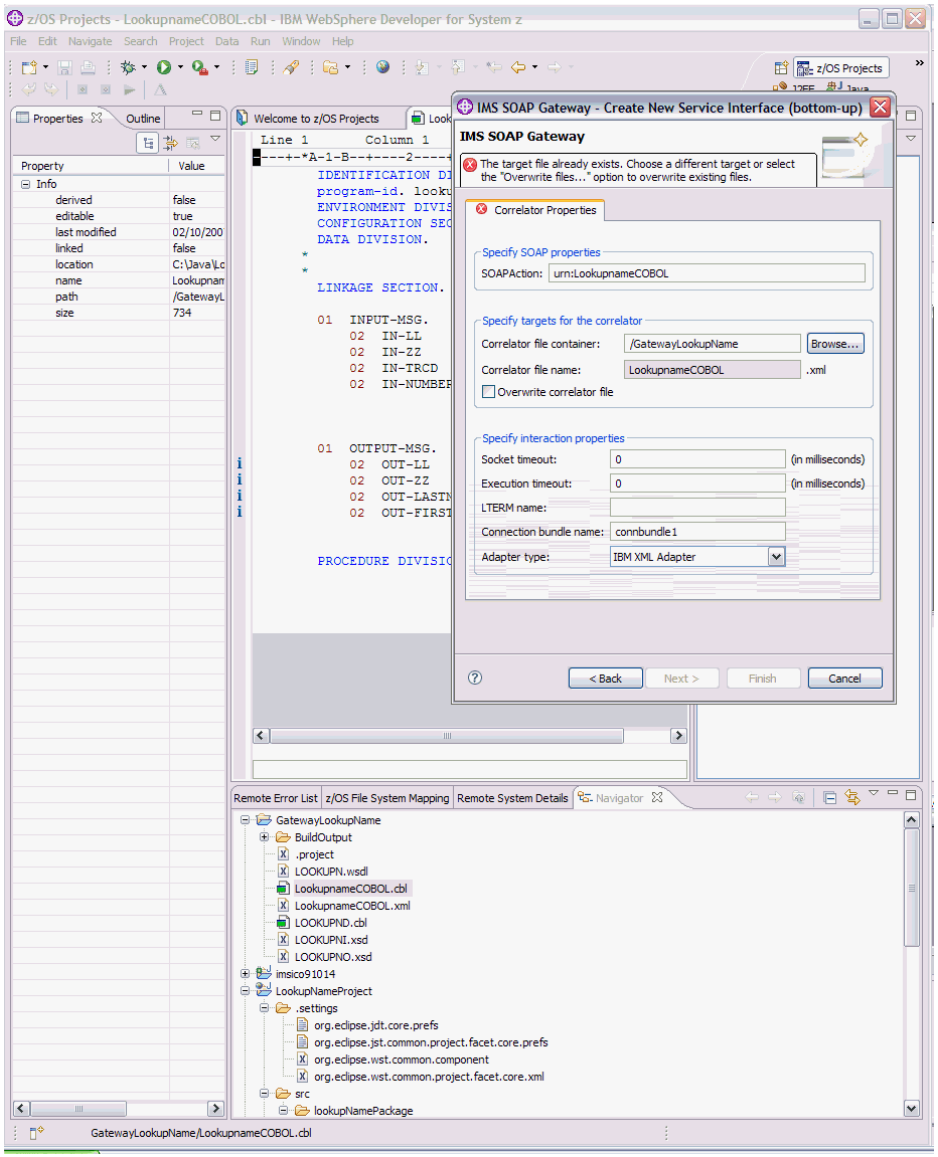
- **IMS SOAP Gateway is a light-weight Web Services server**
  - In place of a Web Application Server
- **Works in conjunction with IMS Connect**
  - IMS SOAP Gateway handles the SOAP Envelope, leaving the input message in XML format
  - IMS Connect calls routines for transforming input XML message into an IMS input message, and vice versa for a reply message
- **Rational Developer for System z (or WDz)† is used to generate the services**
- **Supports Commit Mode 1 with Sync_Level=NONE**



† two free copies of WDz (or RDz) are made available for download by all IMS 10 customers

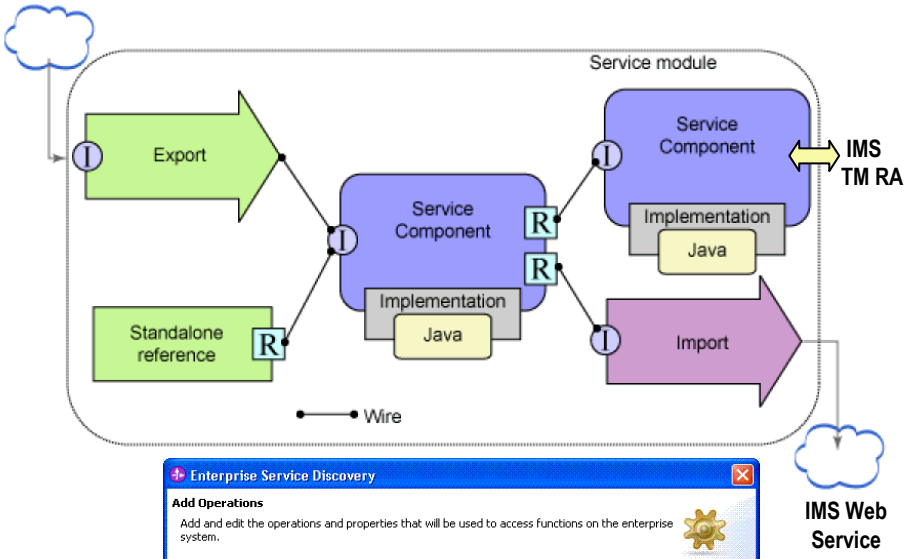
Generating Web Services with RDz for IMS SOAP Gateway

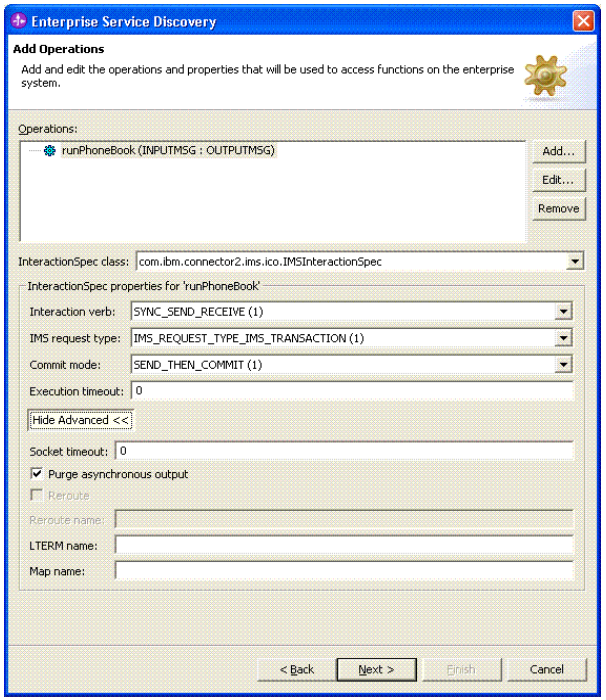
- **Step 1**
  - Import IMS application source code of message structures
  - Use RDz Enterprise Service Wizard to generate
    - WSDL
    - IMS SOAP Gateway Correlator File (XML)
    - COBOL XML Converter (for IMS Connect)
- **Step 2**
  - Compile the XML Converter and link it into IMS Connect STEPLIB dataset
- **Step 3**
  - Use IMS SOAP Gateway Deployment Utility to deploy WSDL and Correlator File to IMS SOAP Gateway



Using WID to Develop SCA Solutions with IMS TM

- **WebSphere Integration Developer is typically used to wire together service components to compose service modules or processes**
- **You can use WID to create a service component that "imports" an existing IMS Web Service (e.g. generated with RAD)**
- **Alternatively, for simple IMS transactions, you can generate a component that directly calls an IMS Transaction via the IMS TM Resource Adapter**
  - GUI interface similar to RAD



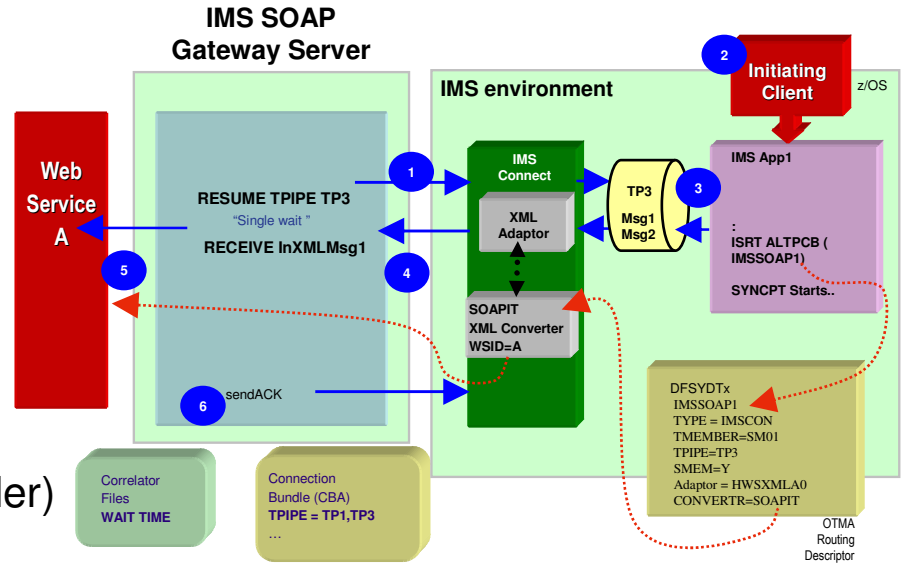


Calling Services from IMS Transactions

- **In SOA, a service can call another service**
- **IMS 10 Callout support allows an ALTPCB call to be used to call an external service. For example -**
  - EJB in WAS
  - Web Service via IMS SOAP Gateway
- **IMS 10 provides features that makes this implementation simple and practical**
  - OTMA Destination Routing Descriptors (instead of routing exits)
  - IMS Connect Resume TPIPE with Alternate ClientID
  - OTMA Resume TPIPE Security
- **This is currently an asynchronous solution**
  - Calling application does NOT wait for response
  - If a response is returned, it should be used to drive a second transaction
- **The requirement for synchronous callout is well understood**

IMS 10 Asynchronous Callout Options

- **With WebSphere Application Server**
  - You write a “listener” EJB (or Message Driven Bean) that waits on messages for an ALTPCB destination
    - The EJB can process a message itself or call a service (EJB, Web Service, etc)
    - To develop this solution, RAD can be used to generate the data bindings for the ALTPCB message, and this will simplify the coding of the EJB
- **With IMS SOAP Gateway 10.1**
  - The SOAP Gateway will act as the Listener, and call the appropriate Web Service for each message received
  - Solution developed with RDz 7.1, which will take the WSDL of the target Web Service and generate –
    - The COBOL structure for the ALTPCB message
    - the XML converter routine for IMS Connect
    - the Correlator file for the IMS SOAP Gateway (used to build IMS Connect header)



Developing New IMS Transactions

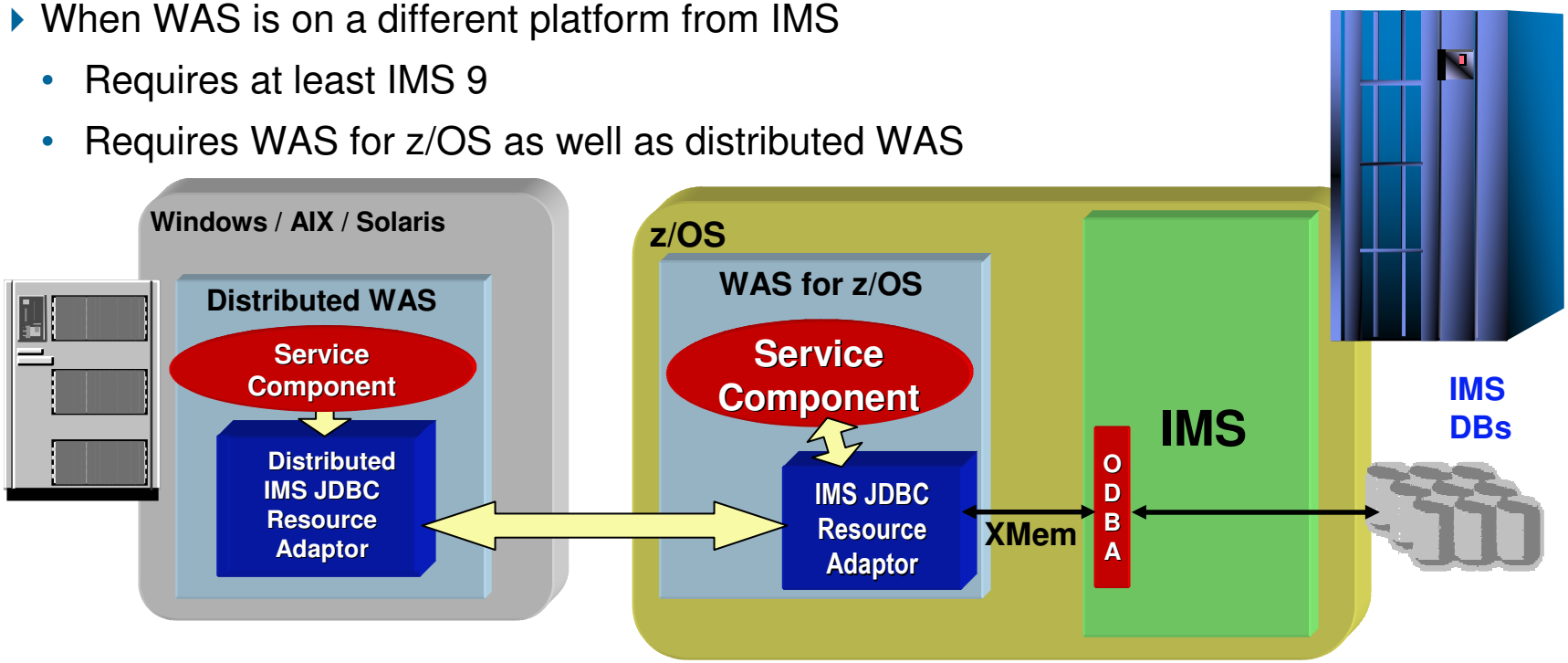
> *The SOA Foundation architecture embraces legacy as a tremendously valuable asset and deliberately avoids requiring that you re-engineer that entire legacy into a new generation of technology or language.*
>
> *… possibly re-factoring it to be more re-usable, and then augmenting it to more precisely match the requirements of the business design.*
>
> IBM's SOA Foundation Nov 2005

- **Service-enabling existing applications may require additional service function be added**
- **In many cases – for example in a distributed environment with multiple IMS DB accesses – the best solution will be to write new transactions**
- **When an existing transaction does not exactly match the business requirement, the most efficient solution is to modify the existing transaction**
  - And the best tool for developing new or modifying existing IMS transactions (in java, COBOL, PL/1, etc) is WebSphere Developer for System z 7.0, or its follow-on, Rational Developer for System z 7.1
- **The alternative is to write a service that runs in WAS and directly accesses the IMS databases**

WebSphere Application Server and IMS DB Access

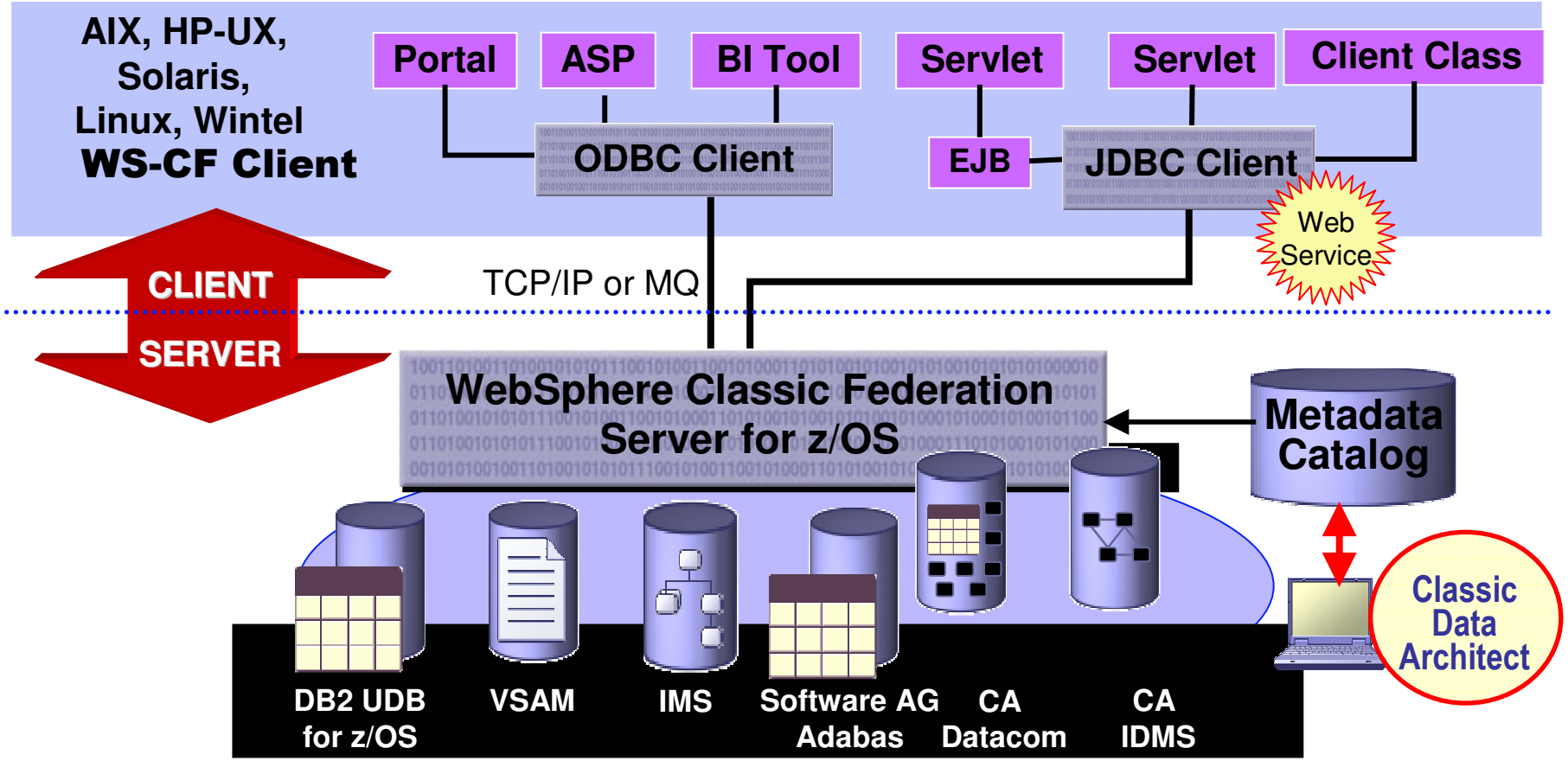
- **External subsystems such as WAS can use IMS's *Open DB Access (ODBA)* to access the online IMS DBs**
- ***Servlets* can use JDBC to access IMS DBs**
  - When WAS and IMS are in same z/OS
- ***EJBs* can use JDBC to access IMS DBs**
  - When WAS and IMS are in same z/OS
  - When WAS is on a different platform from IMS
    - Requires at least IMS 9
    - Requires WAS for z/OS as well as distributed WAS

In other words, services implemented in java can directly access IMS DBs using IMS's own JDBC support



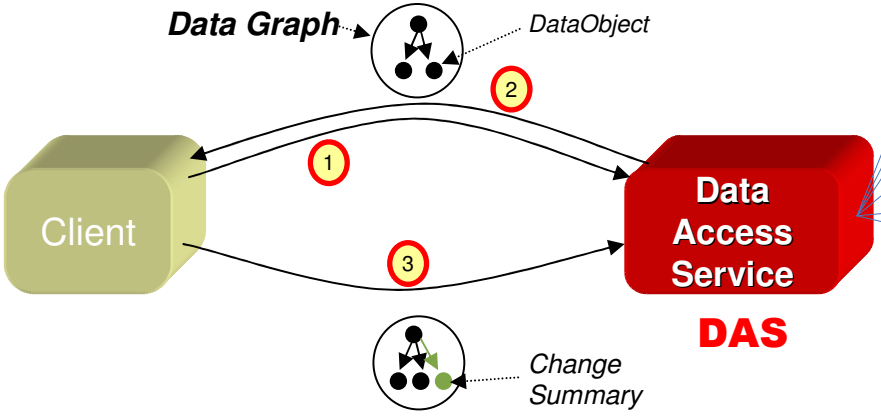
WebSphere Classic Federation Server for z/OS

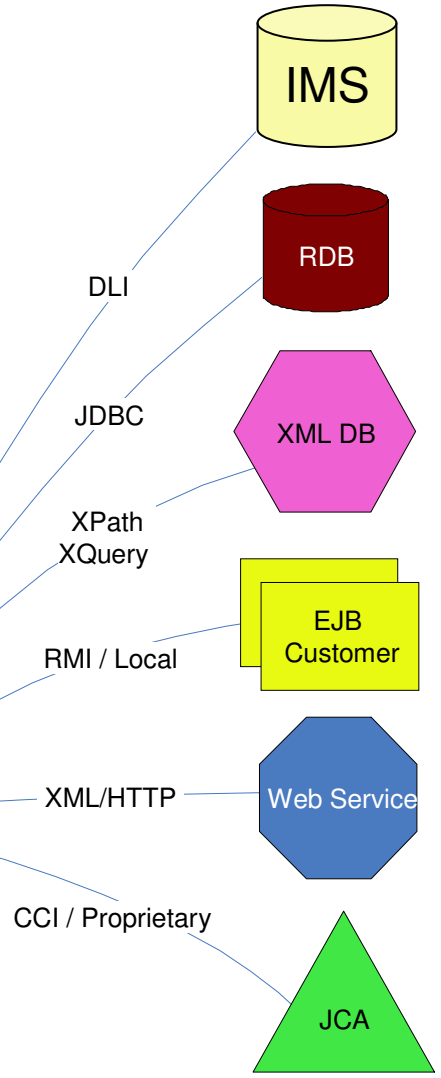
- **More sophisticated access to IMS DBs is possible with WebSphere Classic Federation Server for z/OS**
  - Supports multiple DBMSs (IMS, DB2, etc) including JOINs across them
  - Accesses IMS via ODBA



Service Data Objects with IMS DB

- **There are many technologies for accessing data from the different types of data store – Relational DB, XML DB, Entity EJBs, etc**
  - **Service Data Objects (SDO)** provides a standard data access facility for all data sources in an SCA environment
- **With SDO, client uses a Web Service to send data request to a Data Access Service (DAS) which hides the real DBMS**
- **Client receives back a data graph (or returns it for update)**
  - SDO API used by client to process the data graph
- **Additionally, SCA Modules should (ideally) exchange data using SDOs**
- **IMS 10 will include an IMS Data Access Service**
  - Receive request from client
  - Convert it into IMS DB accesses (DL/1 calls)
  - Return data graph to client





Summary

- **SOA is NOT about rewriting and replacing existing systems**
  - On the contrary, it is about reusing legacy assets in new ways
- **There are various ways of service-enabling IMS transactions**
  - Associating them with MQ queues, or
  - wrapping transactions with EJBs, SCA components, or Web Services
- **IBM's tools – RAD, WID or RDz – make this easy to do**
  - Fully supports the various IMS transaction attributes (including input only, conversational, multi-segment messages, access of asynchronous output, specification of LTERMname, etc.)
- **In some cases the best solution for providing new service function will be to write new or modify existing transactions**
  - The best tool for this is WDz or RDz
- **New java code can be written (servlets, EJBs, SCA components, Web Services) that directly access IMS DBs**
  - Exploiting IMS Java or WebSphere Classic Federation Server for z/OS
  - IMS 10 will enable SDO support for the SCA environment

Thank You for Joining Us today!

Go to www.ibm.com/software/systemz **to:**

- Replay this teleconference
- Replay previously broadcast teleconferences
- Register for upcoming events