E38

IMS Connector for Java: Running J2EE Applications to Access IMS Transactions and Advanced Topics

James D. Polo

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

RETURN TO INDEX

Agenda

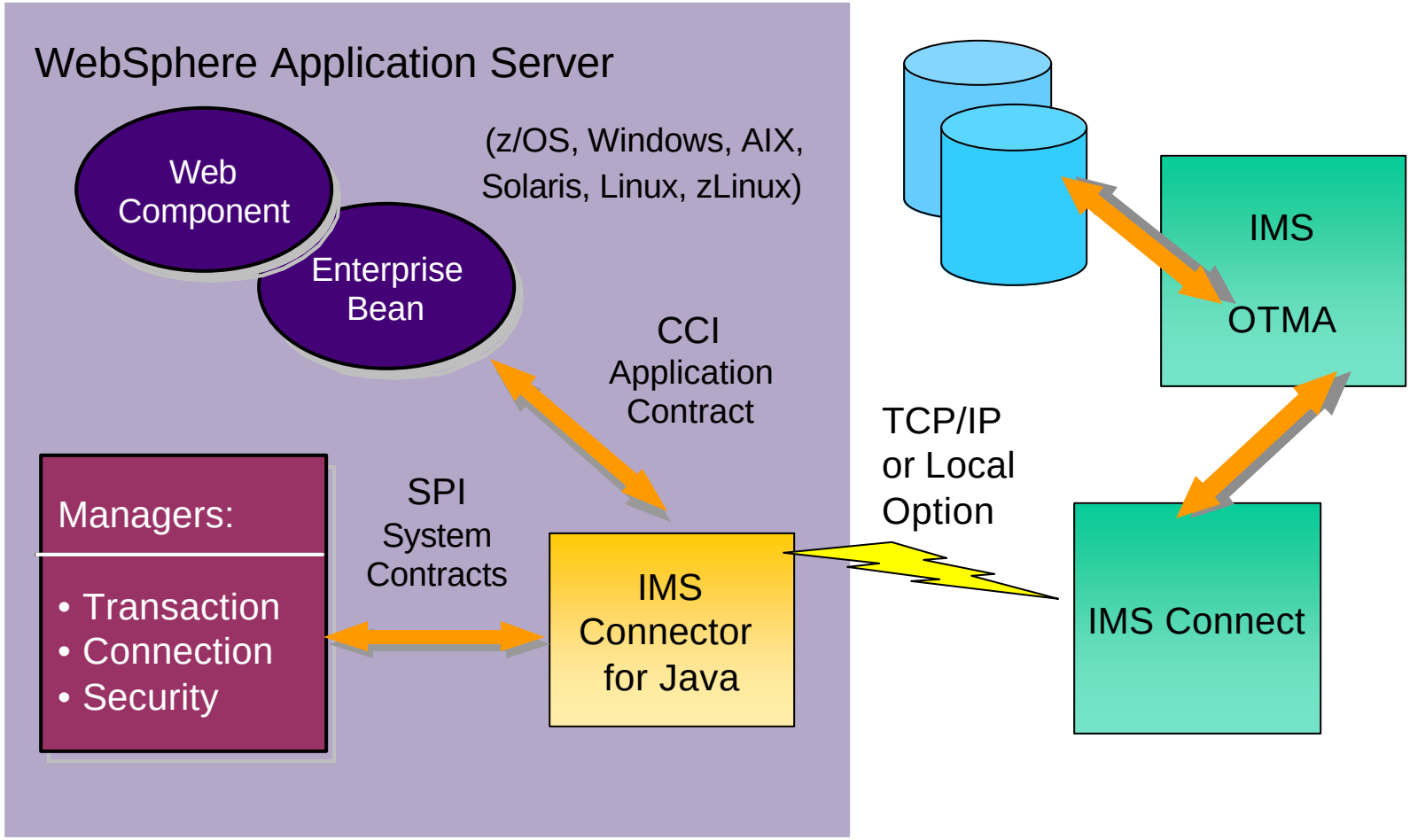
- IMS Connector for Java
- WebSphere Application Server and IMS Connector for Java
  - Application deployment
    - Connection Factory configuration and Topology
  - Quality of Service (QoS)
    - Connection Management
    - Security Management
    - Transaction Management and two-phase commit
- Other features
  - Secure Sockets Layer (SSL)
  - Commit Mode
  - Retrieving asynchronous output messages
  - Execution timeout

IMS Connector for Java

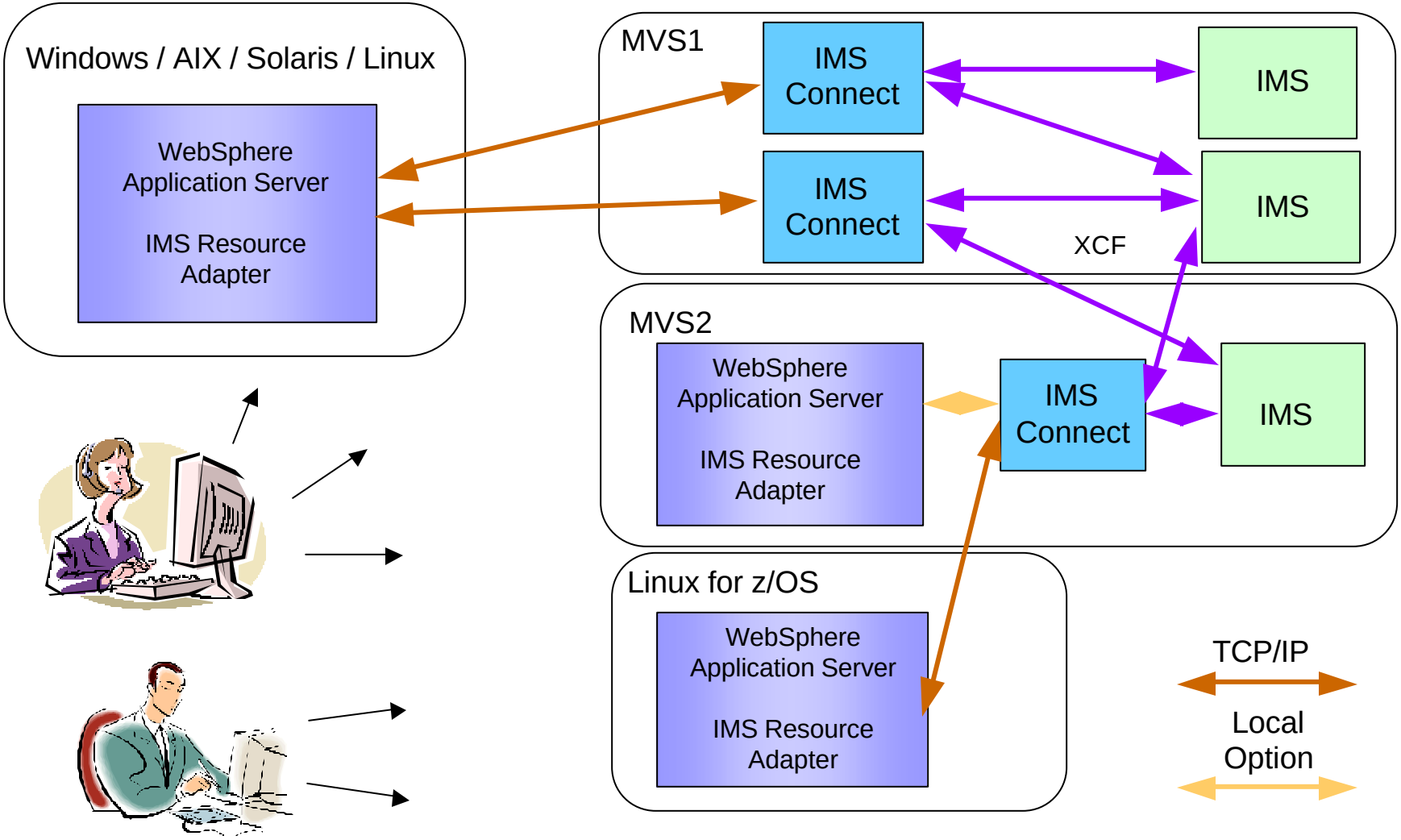
- A J2EE Connector Architecture (JCA) resource adapter
- Used to develop and run J2EE application that access IMS transactions via IMS Connect
- Offers a highly scalable and flexible topology
- Supports rapid application development with WebSphere Studio Application Developer Integration Edition
- Runs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

Helps IMS Users make the transition to ebusiness easier

J2EE Connector Architecture



Topology



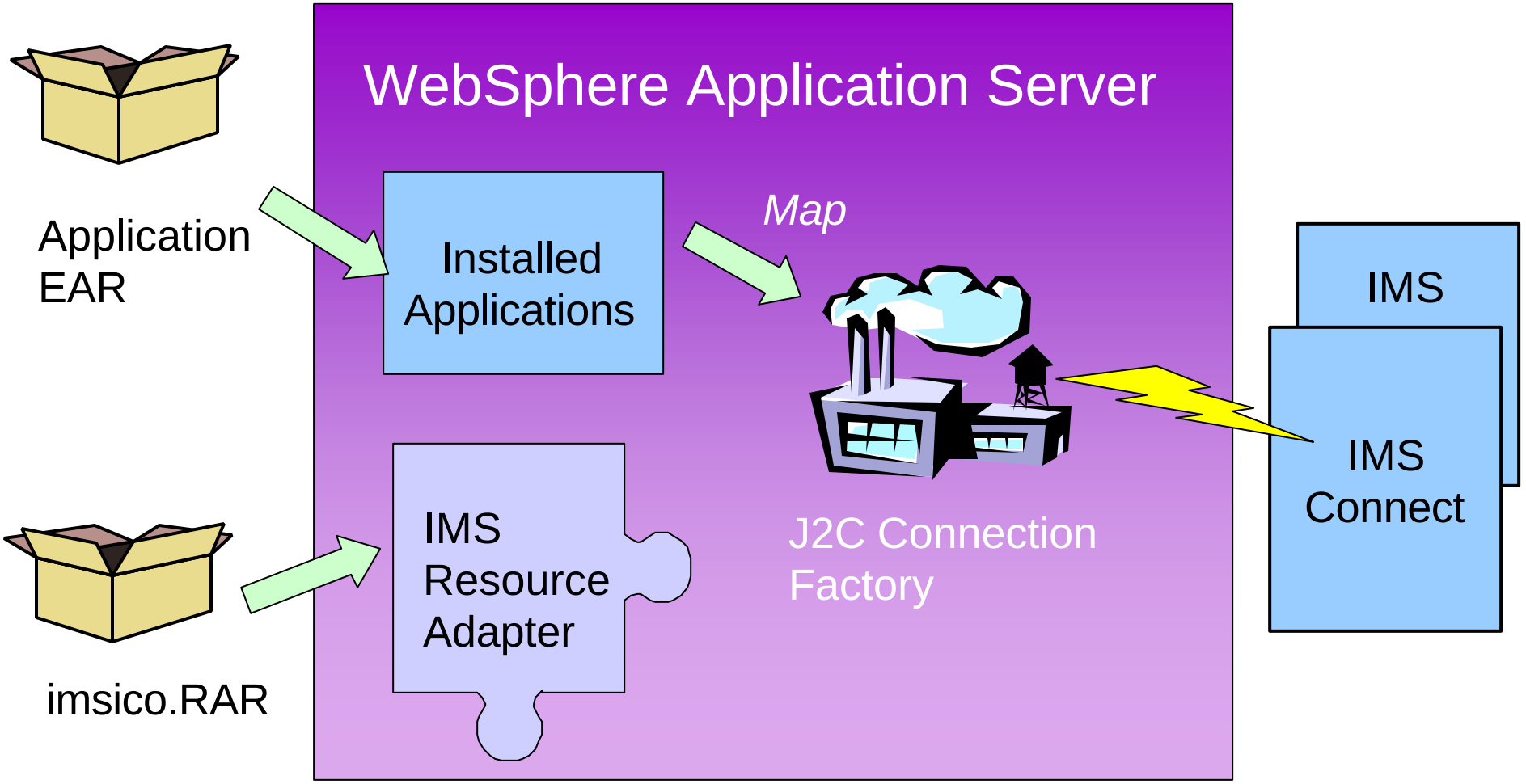
Agenda

- IMS Connector for Java
- WebSphere Application Server and IMS Connector for Java
  - Application deployment
    - Connection Factory configuration and Topology
  - Quality of Service (QoS)
    - Connection Management
    - Security Management
    - Transaction Management and two-phase commit
- Other features
  - Secure Sockets Layer (SSL)
  - Commit Mode
  - Retrieving asynchronous output messages
  - Execution timeout

Deploying to WebSphere Application Server

1. Install the IMS Resource Adapter
   - Resource Archive (RAR) file is included in the files provided with the IMS Connector for Java runtime
2. Configure a J2C IMS Connection Factory
   - Specify a JNDI name
   - Specify custom properties (host name, port, datastore, max connections, etc.)
3. Install the enterprise application (EAR)
   - The Enterprise Archive (EAR) file can be created by **export**ing the application from WSADIE
   - Map the resource reference of the application to the configured connection factory

Application Deployment



J2C Connection Factory

- Provides an application component with a connection to an EIS at runtime
- Configure one or more Connection Factories for the resource adapter
  - Specify the connection and security properties
  - Bind to a JNDI name which can be looked up by an application component
- Application components (e.g., EJBs) have resource references that can be mapped to a specific connection factory
  - Associate the resource reference to the JNDI name of the connection factory when **deploying** (installing) the application

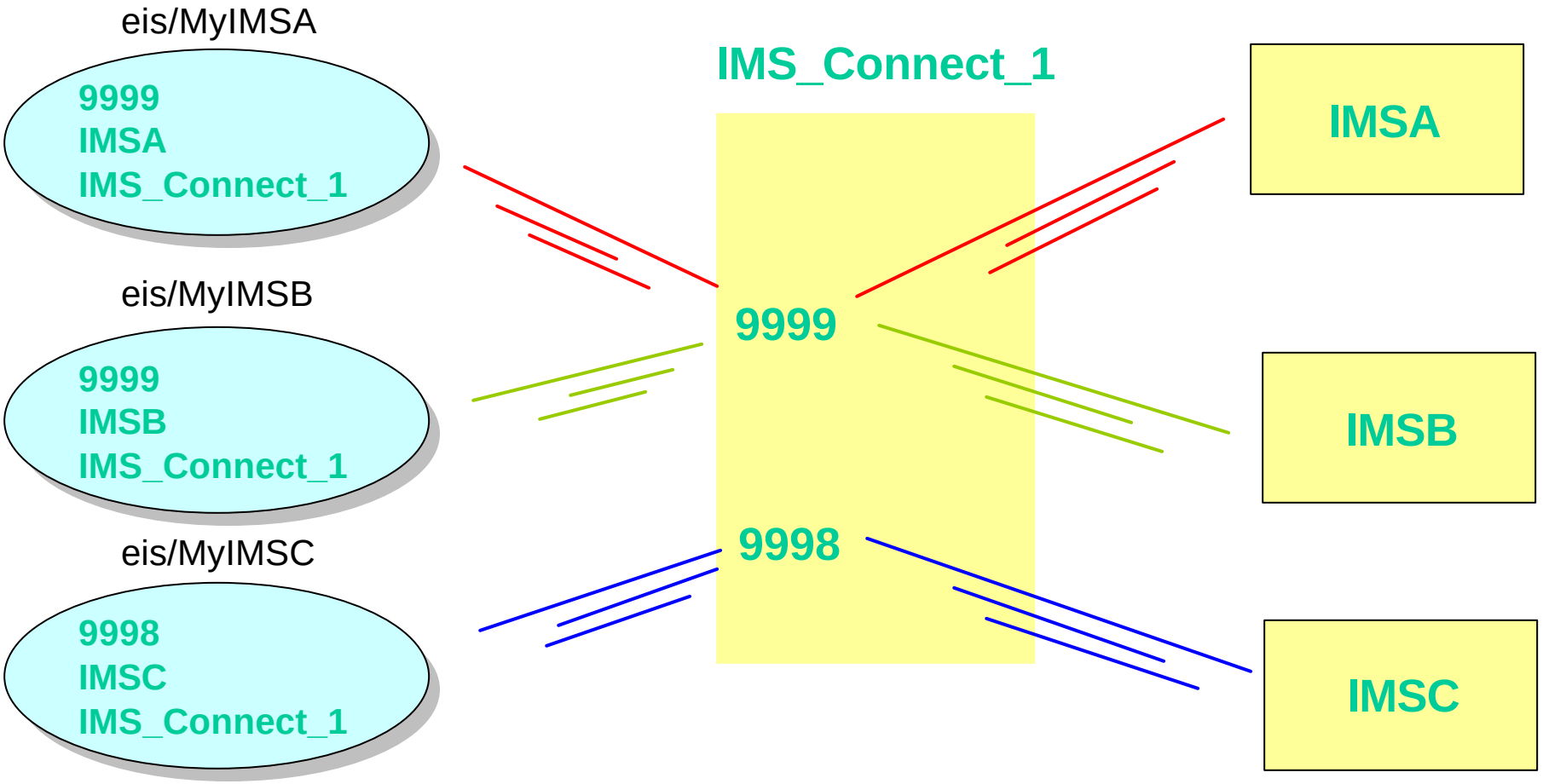
Quality of Service (QoS)

- 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

Connection Management

- Connection Pooling
  - WebSphere Application Server maintains pools of connections
    - For example, 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 Factory



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)
  - The application server (**Container-managed** Sign-on)
- IMS Resource Adapter passes the security information
  - To IMS Connect, as required, for authentication
  - To IMS OTMA for authorization

Component-Managed Sign-on

- The application component provides security information to the resource adapter for EIS Sign-on
  - Set "**res-auth = application**" in the application's deployment descriptor
  - Pass userID, password, and (optional) group name in **IMSConnectionSpec**
  - Default values are used when no IMSConnectionSpec is specified or the userID not specified in IMSConnectionSpec
    - Obtained from Custom Properties of Connection Factory
      - Can override using Component-Managed Authentication alias defined for the Connection Factory
        - » userID and password are associated with the alias

```
IMSConnectionSpec connSpec = new IMSConnectionSpec();
connSpec.setUserName("user1");
connSpec.setPassword("pwd1");

Connection connection = ConnectionFactory.getConnection(connSpec);
```

Container-Managed Sign-on

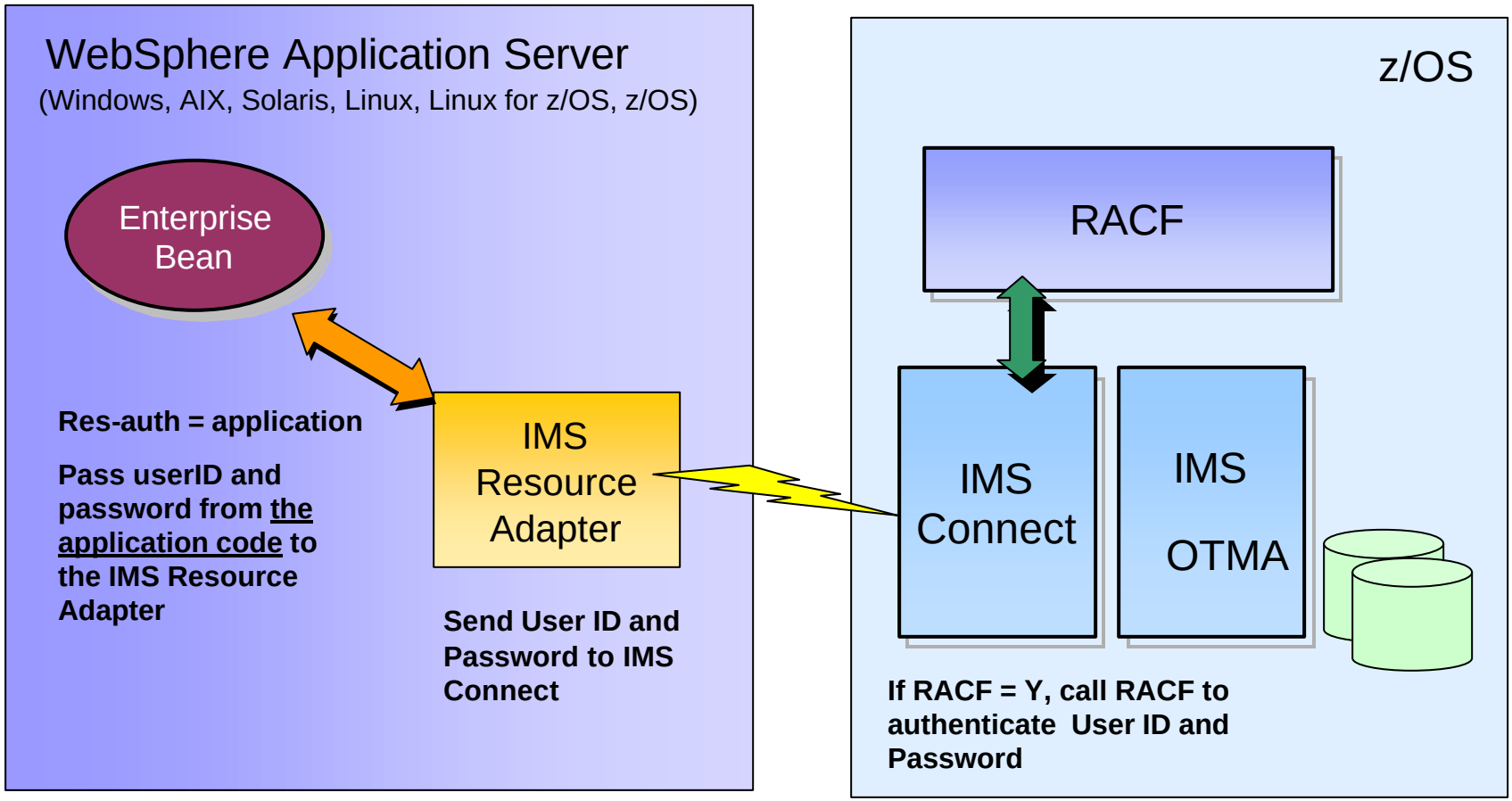
- The container (J2EE Server) provides the security information to the resource adapter for EIS Sign-on
  - Set "**res-auth = container**" in the application's deployment descriptor
  - Provide security information by:
    - Defining Container-Managed Authentication Alias for the Connection Factory
      - userID and password associated with the alias
  - Simplifies application coding - applications are developed without consideration for user authentication issues
  - Security information is centralized and easy to maintain

```
Connection connection = ConnectionFactory.getConnection();
```

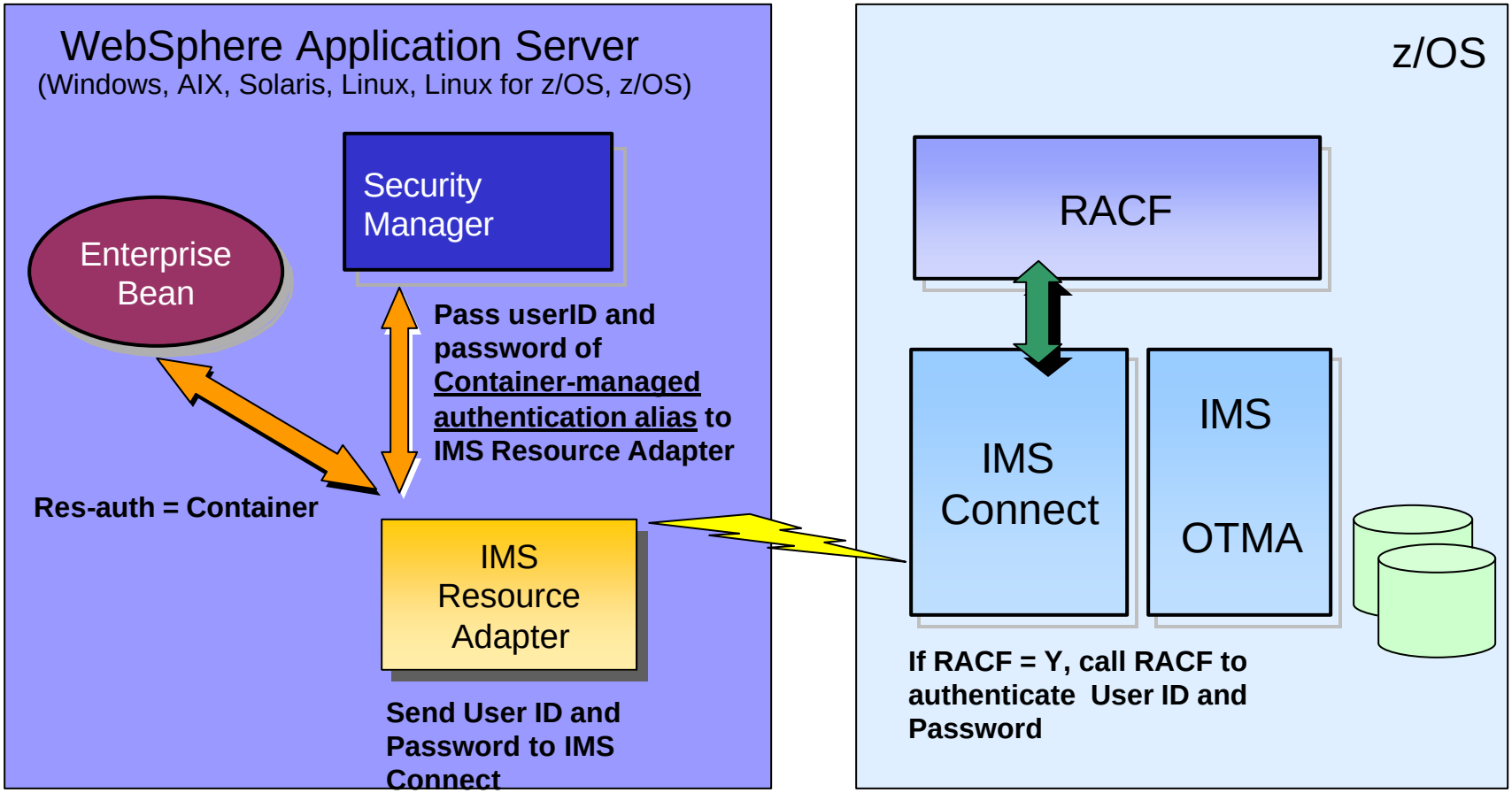
Container-managed with WAS z/OS Thread Identity

- WebSphere Application Server z/OS supports using identity associated with the thread of the application component
- IMS Resource Adapter supports Thread Identity when using Local Option communication
  - WAS z/OS passes the already verified thread identity to IMS Resource Adapter
  - Authentication is bypassed in IMS Connect since authentication has already been performed by WAS z/OS
- To use Thread Identity:
  - Configure your application to use Container-Managed Sign-on (**res-auth = Container**)
  - Configure your application with the desired thread identity using the **Run-As** property in the deployment descriptor
  - Define the Connection Factory to use **Local Option** communication
  - Do NOT specify the Container-Managed Authentication Alias in the Connection Factory

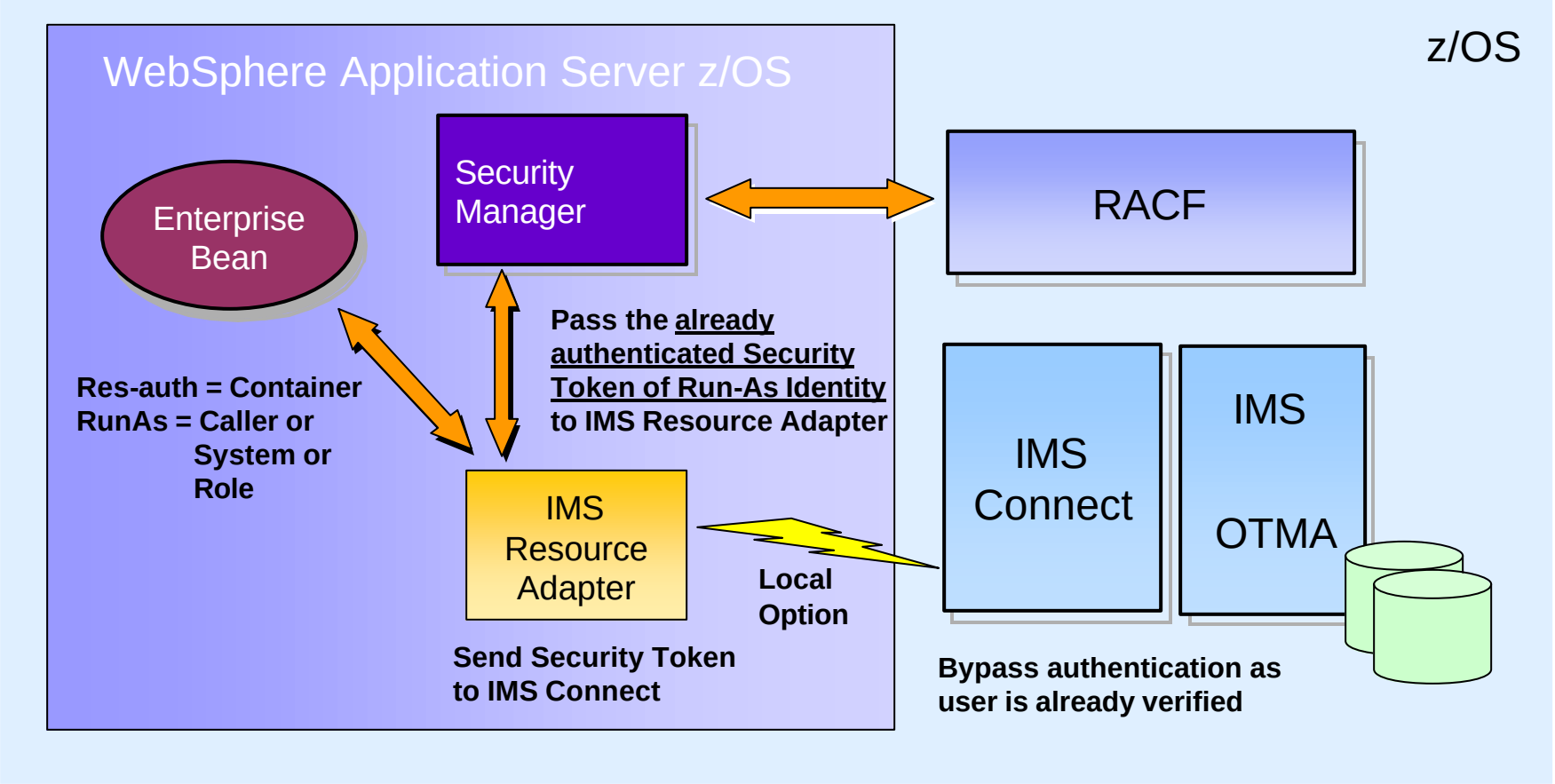
Component-managed Sign-on



Container-managed Sign-on



Container-managed Sign-on with WAS z/OS Thread Identity



Transaction Management

- Transaction (unit of work) is a group of changes that are either fully completed or fully rolled back
- Enables consistent changes to one or more protected resources
- WebSphere Application Server contains a transaction manager that supports the coordination of resource managers using two phase commit
- IMS Resource Adapter supports two phase commit processing
  - RRS transaction with Local Option in WAS z/OS
  - XA transaction with TCP/IP in WAS distributed and z/OS
- J2EE offers two options on how to set the transaction boundaries in your application
  - Bean-Managed transaction
  - Container-Managed transaction

Bean-Managed Transaction

- Can be used in EJB and Servlet applications
- The application code sets the transaction boundaries
- Uses Java Transaction API (JTA) interface to demarcate the transaction

Transaction begins →

Transaction ends →

```
void methodA() {

   java.transaction.UserTransaction transaction =
      ejbContext.getUserTransaction();

   // Indicates the beginning of the group of changes
   transaction.begin();
   :
   // Make changes to the protected resource(s)
   interaction.execute(interactionSpec, input, output);
   :
   // Commit the changes
   transaction.commit();
}
```

Container-Managed Transaction

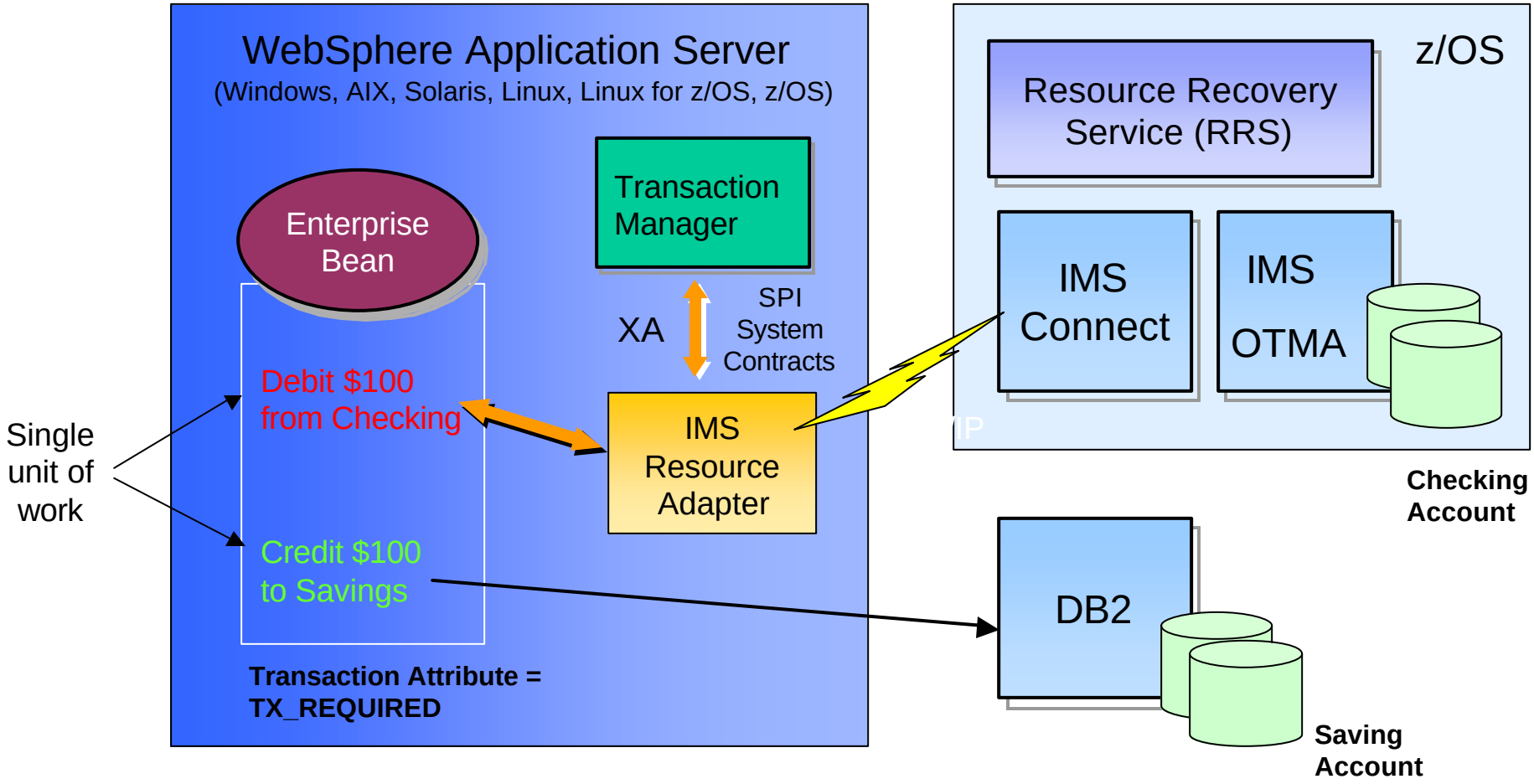
- Can be used in EJB applications
- EJB container sets the transaction boundaries
  - Simplifies application coding
  - Begins a transaction before an enterprise method starts
  - Ends a transaction before the method exits
- Transaction attributes are specified in the deployment descriptor of the EJB

Transaction begins →

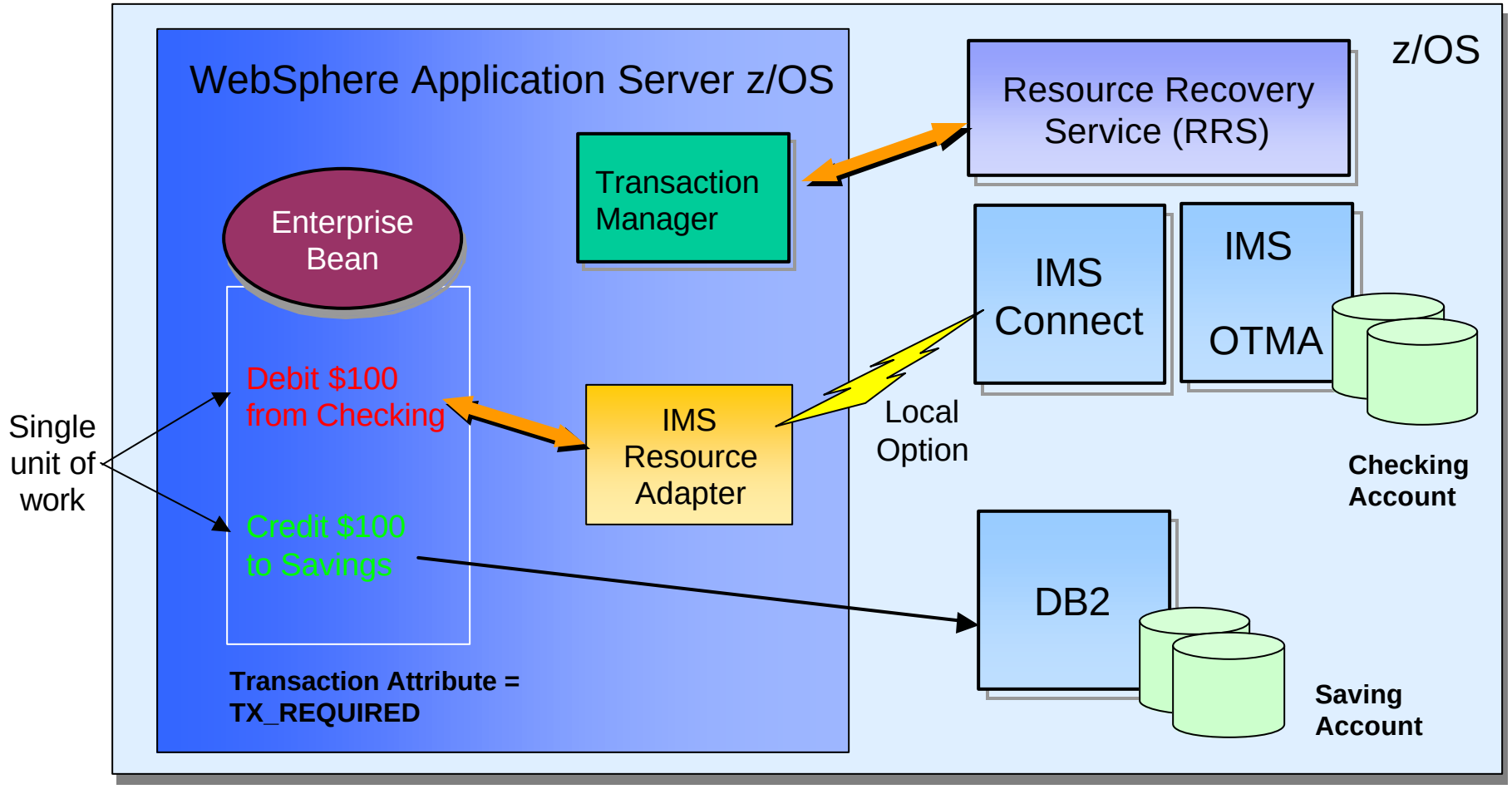
Transaction ends →

```
void methodA() {
:
// Make changes to the protected resource(s)
interaction.execute(interactionSpec, input, output);
:
}
```

Distributed XA Transaction



z/OS RRS Transaction



Considerations when using global transaction

- IMS, IMS Connect and RRS must reside in the same MVS image
  - WAS z/OS must also reside in the same MVS image when using z/OS RRS transaction with Local Option
- MPPs involved in 2-phase commit transactions may require longer wait times
  - Additional MPPs may be needed for workload

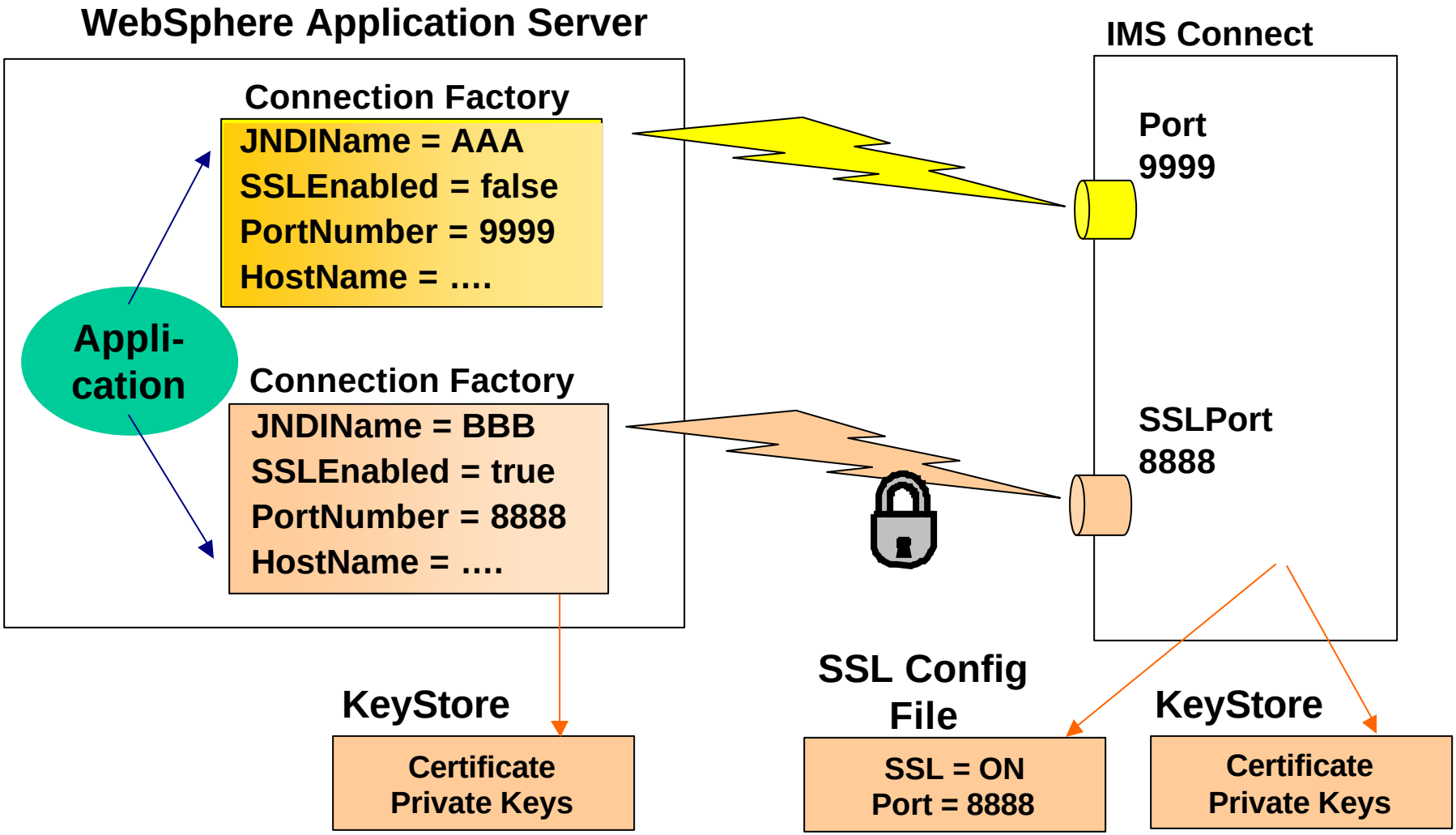
Agenda

- IMS Connector for Java
- WebSphere Application Server and IMS Connector for Java
  - Application deployment
    - Connection Factory configuration and Topology
  - Quality of Service (QoS)
    - Connection Management
    - Security Management
    - Transaction Management and two-phase commit
- Other features
  - Secure Sockets Layer (SSL)
  - Commit Mode
  - Retrieving asynchronous output messages
  - Execution timeout

Secure Socket Layer (SSL)

- Provides Reliable Secure Communication between IMS Resource Adapter and IMS Connect
- Widely accepted as the Industry standard for providing security
- Provides:
  - Server authentication
    - X509 Certificates
  - Client authentication
    - X509 Certificates
  - Data encryption
    - Public and Private Key
  - Message integrity
    - MAC

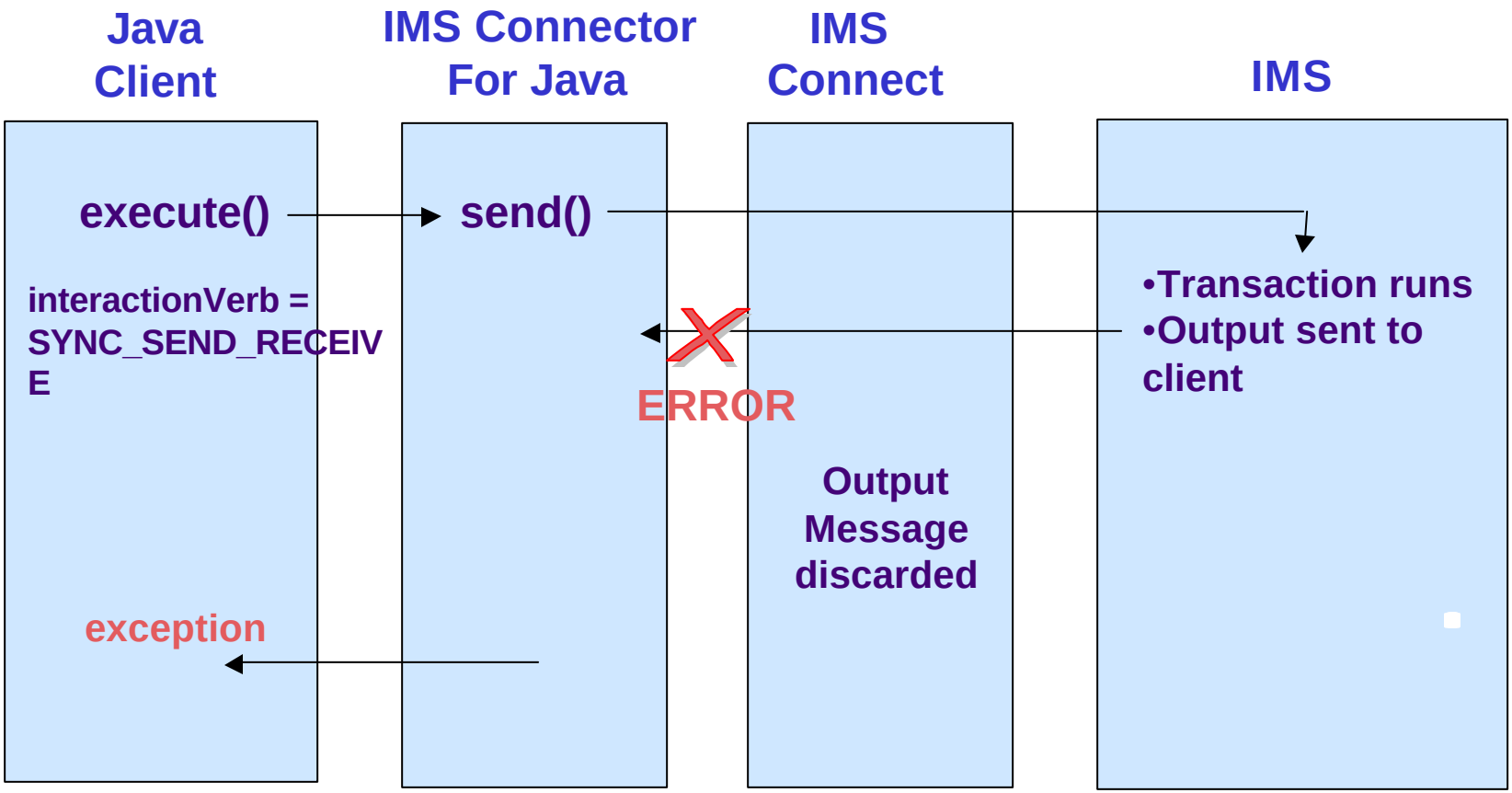
SSL



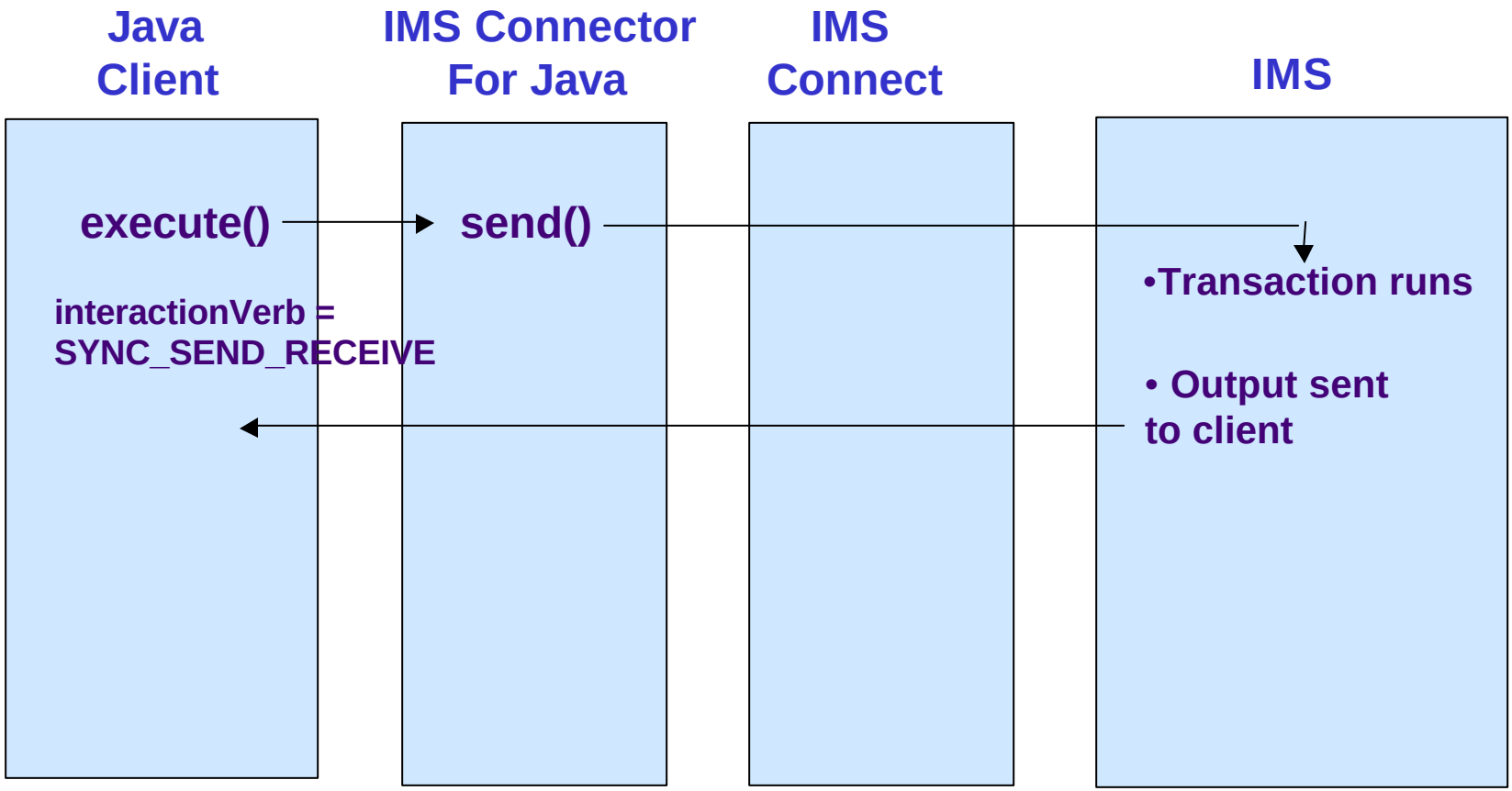
Commit Mode 1 (Send-then-Commit)

- IMS runs transaction; IMS returns output message to client
  - If output message cannot be delivered, it cannot be retrieved later
- Client runs transaction by executing interaction with:
  - IMSInteractionSpec
    - interactionVerb = SYNC_SEND_RECEIVE
    - commitMode = 1
- Internally, IMS Connector for Java uses
    - Persistent socket
    - OTMA syncLevel = None (non-transactional) or Syncpoint (transactional)

Commit Mode 1 Interaction (SyncLevel None)



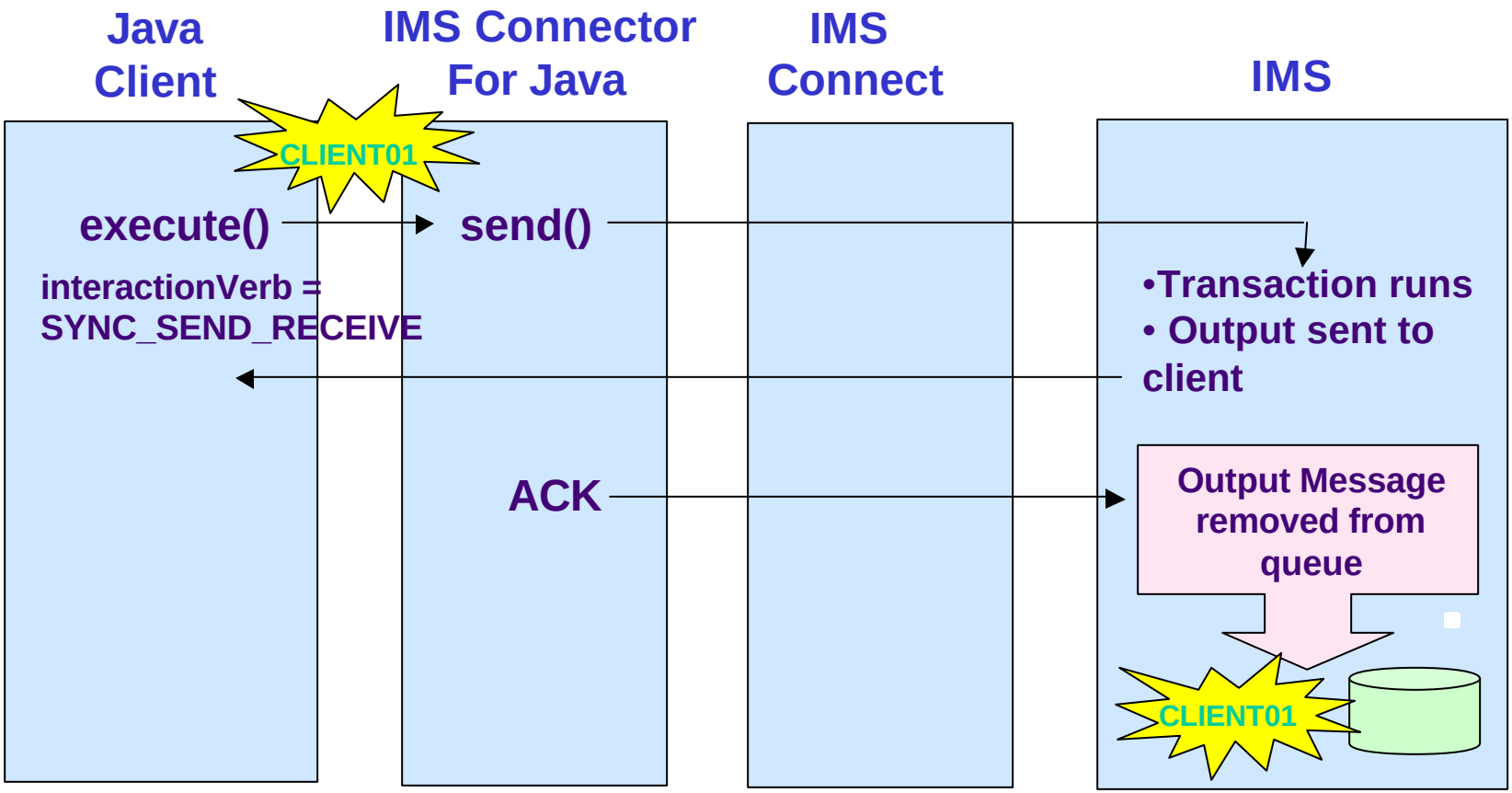
Commit Mode 1 Interaction (SyncLevel None)



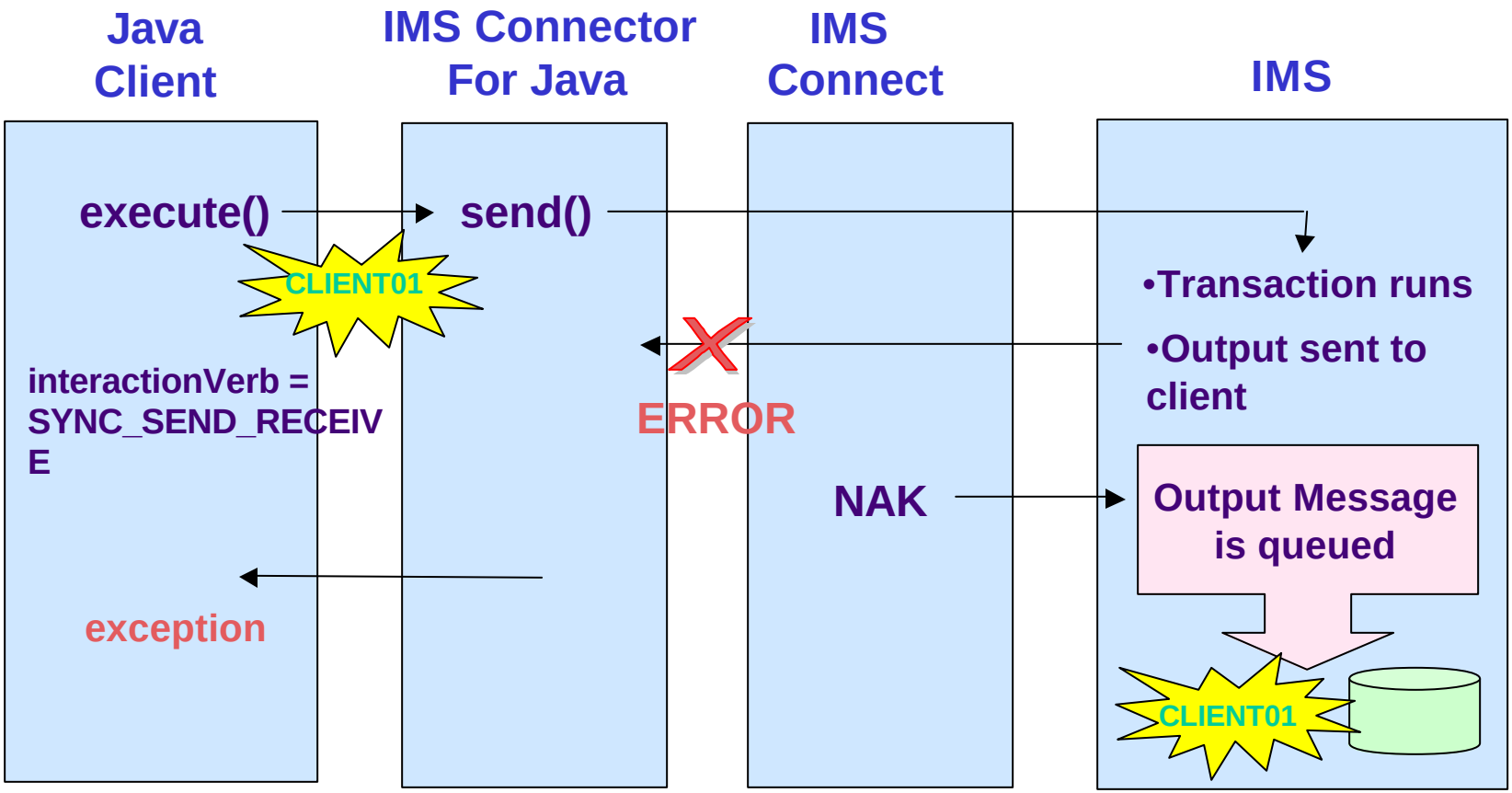
Commit Mode 0 (Commit-then-send)

- IMS runs transaction; IMS returns output message to client
  - If output message cannot be delivered, it can be retrieved later
- Client runs transaction by executing interaction with:
  - **IMSInteractionSpec**
    - interactionVerb = SYNC_SEND_RECEIVE
    - commitMode = 0
  - **IMSConnectionSpec**
    - **clientID** = *user_specified;* must not start with HWS
- Internally, IMS Connector for Java uses
  - Non-persistent socket
  - OTMA syncLevel = Confirm (ACK/NAK sent internally)
- Currently not supported with Local Option

Commit Mode 0 Interaction



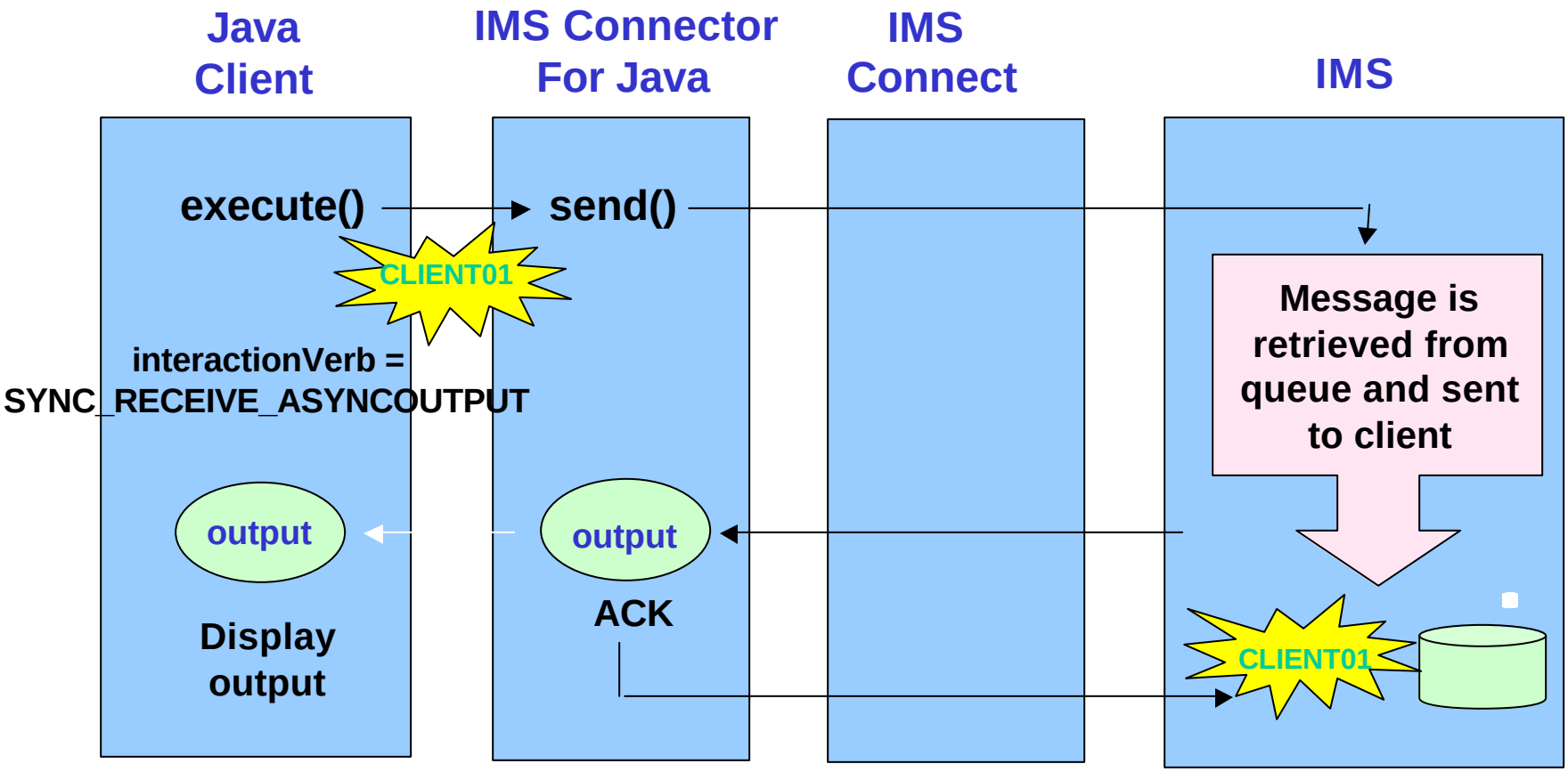
Commit Mode 0 Interaction



Retrieving Asynchronous Output Messages

- IMS enqueues output message to queue (TPIPE) with name **clientID**
  - /DISPLAY TMEMBER *IMSConnect_Name* TPIPE ALL
- Client retrieves message by executing interaction with:
  - **IMSInteractionSpec**
    - interactionVerb = SYNC_RECEIVE_ASYNCOUTPUT
    - commitMode = 0 (default)
  - **IMSConnectionSpec**
    - clientID = the *user_specified* value that was provided on original Commit Mode 0 interaction
- Can also be used to retrieve messages inserted to alternate PCB (**clientID**)
- Internally, IMS Connector for Java uses
  - Non-persistent socket
  - OTMA syncLevel = Confirm (ACK/NAK sent internally)

Retrieve Asynchronous Output



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
- Timeout value is set in the InteractionSpec
  - Valid values are
    - 1 msec to 1 hour
    - 0 to use the timeout value in IMS Connect's config file
    - -1 to wait forever
- Application receives a timeout exception when the timeout occurs and the connection is closed

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/