The J2EE™ Platform Connector Architecture 1.0

Rahul Sharma
Senior Staff Engineer
Sun Microsystems, Inc.
Agenda

• Overview
• Technical details of Connector Architecture
• Status and roadmap
• Q&A
Enterprise Applications

EIS Tier

Middle Tiers

Clients

Enterprise Services

DBMSs

ERP Systems

Device
EIS Integration

• Key issues in EIS integration
  – Heterogeneous and complex EISs
  – Ease of integration and application development
  – Tools
  – Transactions and security
  – Scalability

• Connector Architecture addresses this under the scope of the J2EE 1.3 platform
J2EE™ Platform

• The J2EE platform provides:
  – Faster solution delivery time to market
  – Re-usable components
  – Write Once, Run Anywhere™
  – Huge Industry Support

• Connector architecture adds:
  – Easy EIS connectivity
  – Enterprise Application Integration
Connector Architecture

- Standard architecture for EIS integration
- Targeted EISs under 1.0:
  - ERP systems
  - Mainframe transaction processing systems
  - Non-Java legacy applications
  - Database systems
- Defined through the Java Community Process℠ initiative
High Level Architecture

J2EE Application Server
- Connection Pooling
- Transaction Manager
- Security Manager

Container-Component Contract

Application Component

Resource Adapter

System Contracts
- Transaction Management
- Connection Management
- Security Management

EIS-specific interface

Enterprise Information System
Value Proposition

- Reduces scope from $m \times n$ to $m + n$

- Simplifies application development

- Provides scalable, secure and transactional integration
System Contracts

• System Contracts specified in 1.0:
  – Connection management
  – Transaction management
  – Security management

• Proposed enhancements in later versions:
  – Thread management
  – JMS pluggability
Connection Management Contract

- Supports connection management
  - Connection pooling
  - Configuration of connection factory
  - Creation of connection
  - Matching of pooled connections

- Enables application server to provide Quality of Services (QoS)
Transaction Management

- Local Transaction
  - Managed internally by an EIS resource manager

- XA Transaction
  - Spans across multiple EIS resource managers
  - Requires transaction coordination by an external Transaction Manager
    - Two-phase commit (2PC)
    - One-phase commit (1PC) optimization
Resource Manager (RM) can be:
- Non-transactional
- Local transaction only
- Local and XA transaction

Resource Adapter implements:
- LocalTransaction interface
- JTA XAResource interface

Application server required to support all three transaction levels
Security Management

- Extends the J2EE platform security model for secure EIS connectivity
- Security mechanism and technology independent:
  - Basic user-password mechanism
  - Kerberos v5
  - EIS specific security mechanism
Common Client Interface

J2EE Application Server
- Connection Pooling
- Transaction Manager
- Security Manager

Application Component
- Application Contract

Resource Adapter
- EIS-specific interface
- Enterprise Information System

Container-Component Contract
System Contracts
Rationale for CCI

• Problem for development tools and EAI frameworks:
  – EISs support different client APIs
  – Need for adapting client APIs
• CCI solves this problem
Overview of CCI

- Provides simple remote-function call API
- Focuses on toolability
- Leverages JavaBeans™ architecture-based components and Collections
- Targets EAI and application development tools
- Recommended for resource adapters in version 1.0
Scenario: EAI Framework

Enterprise Application Integration Framework

Common Client Interface

Resource Adapter

JDBC 2.0

JDBC 2.0 Driver

Repository
Scenario: CCI Usage

// Get a Connection
javax.naming.Context nc = new InitialContext();
ConnectionFactory cf =
    (ConnectionFactory)nc.lookup("...");
Connection cx = cf.getConnection();

// Create an Interaction
Interaction ix = cx.createInteraction();

// Create input and output Record
RecordFactory rf = //.. get a RecordFactory
MappedRecord input = rf.createMappedRecord("...");
IndexedRecord output = rf.createIndexedRecord("...");

// Create/get an InteractionSpec
InteractionSpec ixSpec = // ...
ixSpec.setFunctionName("<NAME-OF-EIS-FUNCTION>");

// Execute the Interaction
boolean ret = ix.execute(ixSpec, input, output);
Packaging of Resource Adapter

• Standard packaging format for resource adapter
  – Packaged as JAR
  – Equivalent to "J2EE module"

• XML based deployment descriptor
  – General information
  – Configurable properties
  – Transaction support level
  – Security related configuration
Deployment of Resource Adapter

- Created by Resource Adapter Provider
- Processed by Deployer
- Operational Environment Configured
- Packaged with other J2EE Modules
- J2EE Container
Status and Roadmap

• Defined through Java Community Process initiative
• Release Schedule:
  – Public draft 1: Released 06/2000
  – Proposed Final Draft 1: Released 10/2000
  – Final release: Q2 CY-2001
• Connector.next process initiation
  – Targeted for 02/2001
Reference Implementation and CTS

• J2EE™ RI with Connector support:
  – Early access released in 10/2000
  – Beta release J2EE 1.3 RI: Q1-2001
  – Subsequent RI and CTS releases tied with the J2EE 1.3 platform

• Connector related J2EE RI features:
  – System contracts on container side
  – Packaging and deployment
  – CCI-based sample resource adapter
Connectors 2.0

• Proposed features:
  – Asynchronous resource adapters
  – JMS Pluggability
  – XML support in application contract
  – Metadata support

• Schedule for 2.0:
  – JSR targeted for 02/2001
  – Call for experts
Industry Support

- Part of the J2EE 1.3 platform
- Expert group for Connectors 1.0:
  - BEA, Fujitsu, IBM, Inline, Inprise, iPlanet, Motorola, Oracle, SAP, Sun, Sybase, Tibco, Unisys
- Huge Java™ technology community interest:
  - Successful early access release
  - Strong support for resource adapter providers
Advantages of Architecture

- Application developers
- Application server providers
- Enterprise Information system vendors
- Application development tools provider
- Enterprise application integration (EAI) vendors
- Third-party ISVs
Pointers and References

• Web resources:
  – http://java.sun.com/j2ee/
  – http://java.sun.com/j2ee/connector/

• Spec lead: rahuls@eng.sun.com
Java Message Service (JMS) 1.0

Rahul Sharma
Senior Staff Engineer
Sun Microsystems, Inc.
Agenda

• Overview of the Java™ Message Service API (JMS)
• JMS Reference Implementation
• Preview of J2EE™ platform-based component messaging
• Future considerations
What Is the JMS API?

- A common Java™ platform API for creating, sending, receiving and reading messages
- Enables communication that is
  - Loosely coupled
  - Reliable
  - Asynchronous
- Designed by Sun and partners
- Released 11/1999
JMS API Objectives

• Capable of mapping to existing Message Oriented Middleware (MOM) systems
• Enough functionality to support sophisticated messaging applications
• Enable the development of efficient JMS Providers
• Allow portability of a new JMS application across JMS products in same message domain
JMS Functionality

• Two messaging domains
  – Point-to-Point (Reliable Queue)
  – Publish/Subscribe

• Message delivery
  – Synchronous or asynchronous
  – Reliability provided by acknowledgements

• Message selectors
• Transactions
• Choice of five Message types
Point-to-Point Messaging

- Only one consumer of queue message
- No timing dependencies between sender and receiver
Publish/Subscribe Messaging

- Broadcast message to all subscribers
Vendor Implementations

- 13 implementations of JMS API:
  
  (Note: Listing by vendor request, not all-inclusive)
  
  BEA Systems, Inc.   Fiorano Software, Inc.
  IBM           objectCube, Inc.
  Oracle Corporation      Orion
  Progress Software    Saga Software, Inc.
  Softwired, Inc       SpiritSoft, Inc.
  Sun Java Message Queue
  Valto Systems       Venue Software
JMS Within the J2EE™ Platform

• Enables J2EE components to
  – Interact via first class distributed computing paradigm of message passing
  – Interact with message-enabled legacy systems

• Uses an open standard API
  – Will allow freedom of choice among JMS Providers
Accessing JMS From an EJB™ Architecture-Based Component ("EJB Component")

- Any EJB component type can send or synchronously receive a message
- Message-driven Bean enables asynchronous invocation mechanism
- Msg send and receive can participate in JTA transaction
- Described in the EJB 2.0 specification
Goals of Message-Driven Bean

- As simple to write as any other JMS MessageListener
- Allow for asynchronous concurrent message consumption
Message-Driven Bean Instance

- Executes on receipt of a JMS message
- Shares the following characteristics of stateless session bean
  - Can be transaction-aware
  - May update shared data in an underlying database
  - Is stateless and relatively short-lived
  - Notable difference: No home or remote interface
Overview of EJB Architecture

Messaging Source Code Example

Container

Publisher Bean

Publishes

EJB Server

Subscribes

Msg-driven Instances

Delivers

Acknowledges

Container

Subscriber

Stores

Database

Client App

Durable Subscription
import javax.ejb.*;
import javax.naming.*;
import javax.jms.*;

public class PublisherEJB implements SessionBean {
    TopicConnection tconn = null;
    Topic topic = null;

    // Cache lookup of JMS Connection and Destination
    public void ejbCreate() throws RemoteException {
        Context ctx = new InitialContext("java:comp/env");
        topic = (Topic)ctx.lookup("jms/MyTopic");
        TopicConnectionFactory tcfac = (TopicConnectionFactory)
            ctx.lookup("jms/MyTopicConnectionFactory");
        tconn = tcfac.createTopicConnection();
    }

    // Release JMS connection
    public void ejbRemove() throws RemoteException {
        if (tconn != null) tconn.close();
    }
}
/**
 * Publishes a message to a topic.
 */

public void publishNews() throws EJBException {

    // Create JMS context to publish message to topic
    TopicSession tsess = tconn.createTopicSession(true, 0);
    TopicPublisher tpub = tsess.createPublisher(topic);

    // Create and send message
    TextMessage message = tsess.createTextMessage("News item");
    tpub.publish(message);

    // Release JMS resources
    tsess.close();
}

public class SubscriberMsgBean implements MessageDrivenBean {

    < define ejbCreate(), ejbRemove() and 
    setMessageDrivenContext(MessageDrivenContext mdc) >

    /* Message acknowledge and database update participate 
    * in distributed txn. 
    */
    public void onMessage(Message inMessage) {
        TextMessage msg = (TextMessage) inMessage;

        try {
            < look up JDBC database >
            < store info from message in database >
        } catch(Exception e) {
            e.printStackTrace();
        }
    }
}
Benefits of JMS Within an EJB Component

• Simplify enterprise development with
  – Loosely coupled, reliable, asynchronous interactions between EJBs and other components
  – Ease of extensibility of business events
    • Add a new message-driven bean to introduce new business logic for existing business events

• JMS functionality enhanced by EJB container architecture
  – Distributed transaction support
  – Concurrent consumption of messages
Conclusions

- The J2EE™ platform is ideal for enterprise development
- The Java™ Message Service API (JMS) enables asynchronous, loosely coupled, reliable communication among clients
- JMS adds messaging paradigm to J2EE platform
- J2EE container architecture enhances JMS functionality
More Information

• Web sites
  – JMS: http://java.sun.com/products/jms
  – J2EE: http://java.sun.com/products/j2ee

• Mailing list
  – jms-interest@java.sun.com
  • Join at http://archives.java.sun.com
EAI Scenario

- Order Management Application
- Application Server
- Enterprise Messaging System
- Fulfillment Service Provider
- Customer Database
- Order Database
- Inventory Management Application
  <Mainframe system>
- Logistics Application
  <ERP system>
- Financial Application
  <ERP System>
- Buyer
- eMarketplace

Sun Microsystems
The J2EE™ Platform

Client Tier

Web Server
JSP, Servlet, HTML, XML

Middle Tier

Application Server

Enterprise JavaBean

Enterprise Information Systems (EIS):
Relational-Database, Legacy Applications, ERP Systems

Other Services:
JNDI, JMS, JavaMail™

Client Tier

EIS Tier

Firewall
Using J2EE