

SCALABILITY SECURITY

**Borland**<sup>®</sup>

EJB<sup>™</sup>

RELIABILITY

Web Services

J2EE<sup>™</sup>

# Asynchrone Kommunikation mit Message Driven Beans

**Arnold Senn**  
(Technical Consultant)

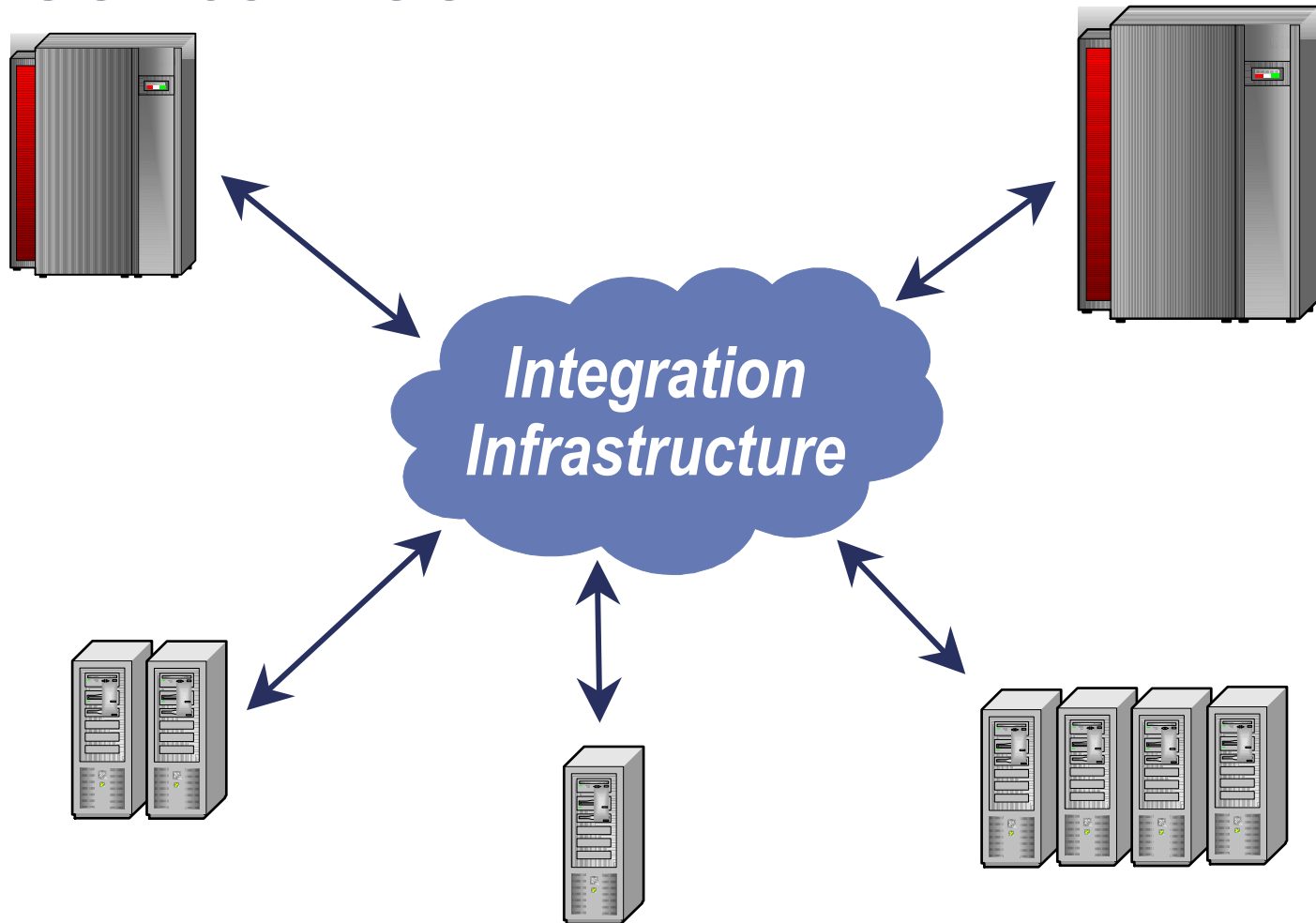
[asenn@borland.com](mailto:asenn@borland.com)

**Borland**

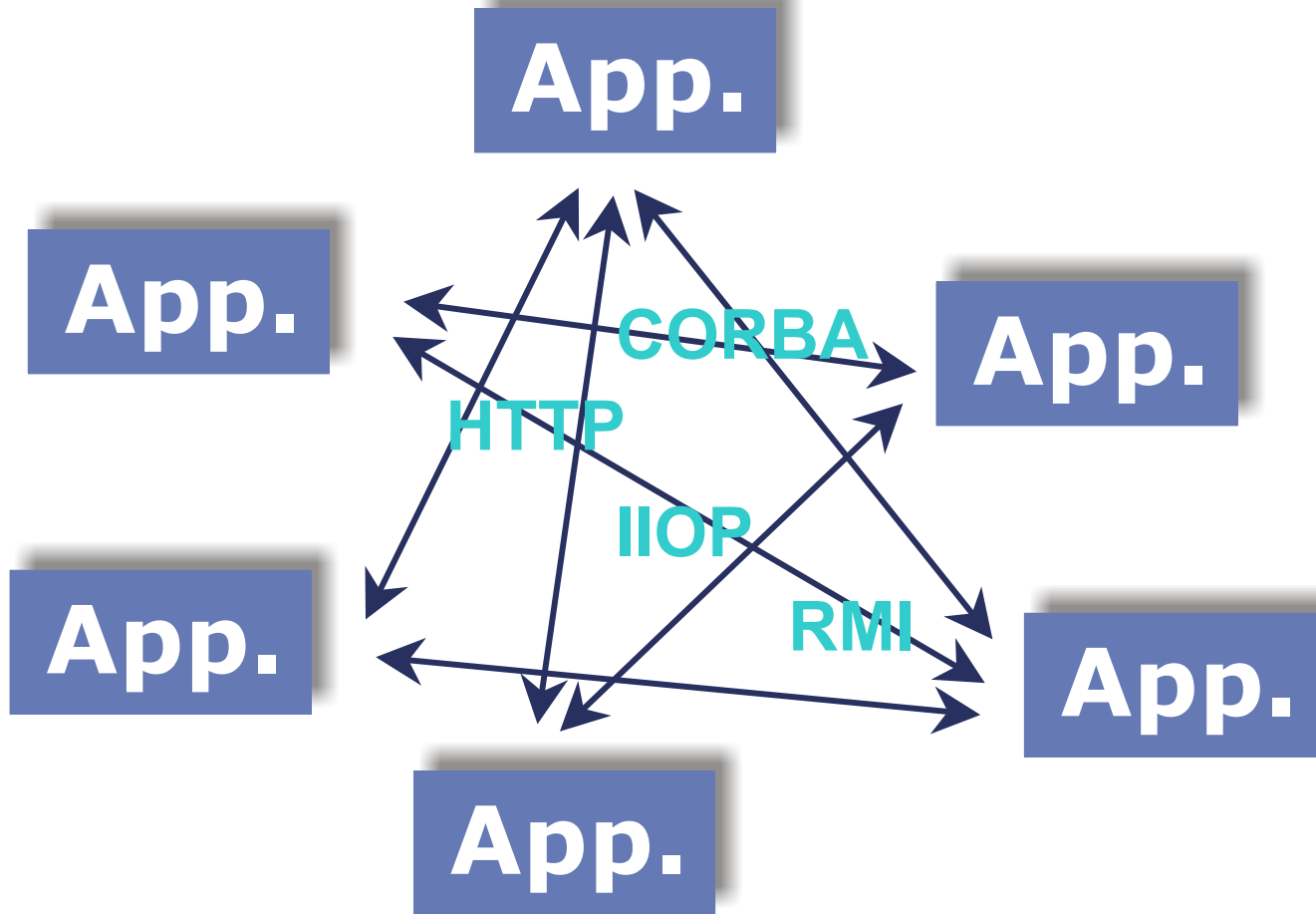
# Outline

- **Why Messaging Systems?**
- **Concepts**
- **JMS specification**
  - Messaging Modes
  - Messages
  - Implementation
- **EJB 1.1 and JMS**
- **EJB 2.0 with Message Driven Beans**
- **Message Driven Beans**
- **Why Message Driven Beans?**

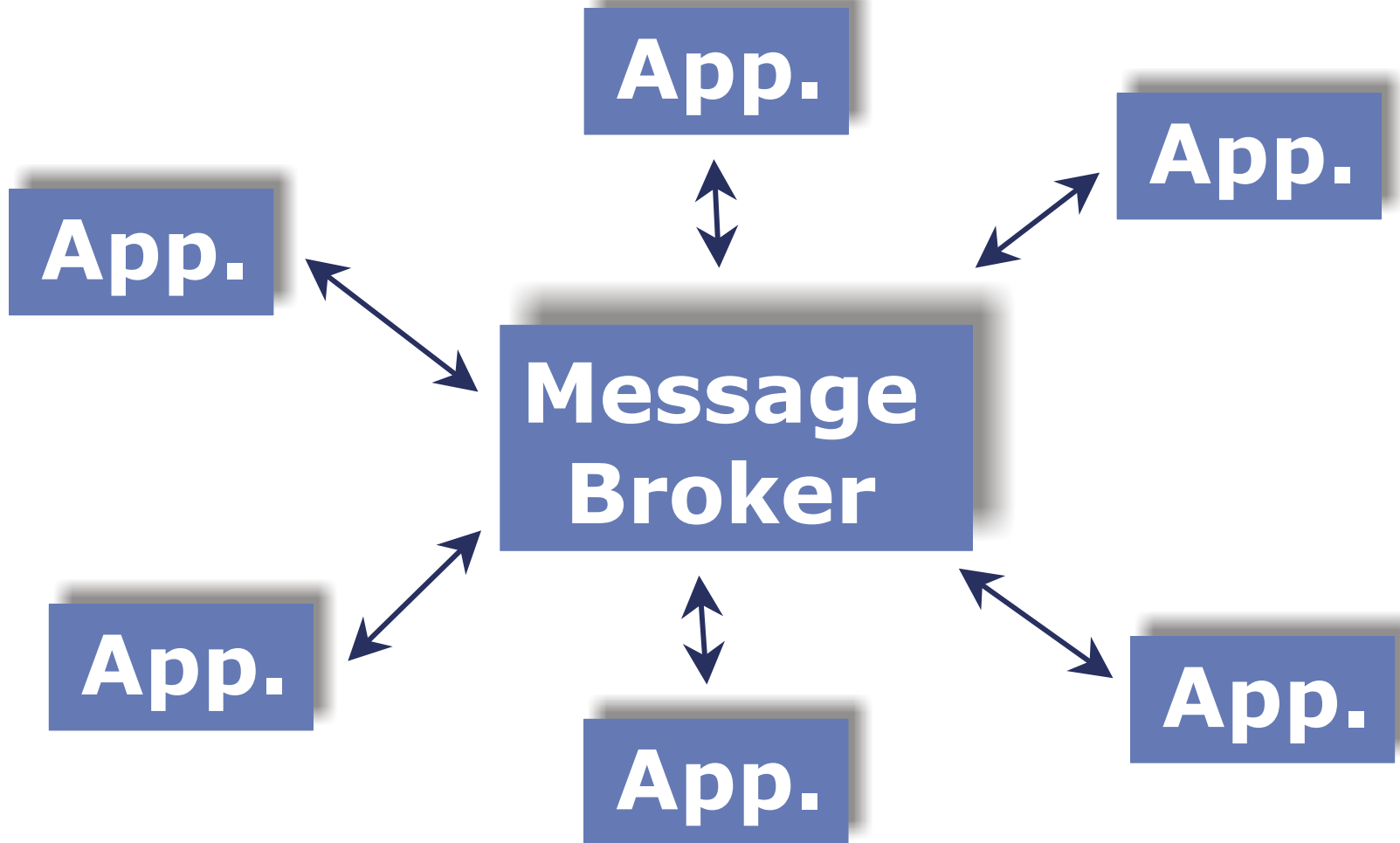
# Peer to Peer



# Peer to Peer



# Hub & Spoke



# Why Message Middleware?

- **Single point of integration**
- **Loosely-coupled integration**
- **Simplified Integration of different platforms and applications**
- **Asynchronous communication**

# Outline

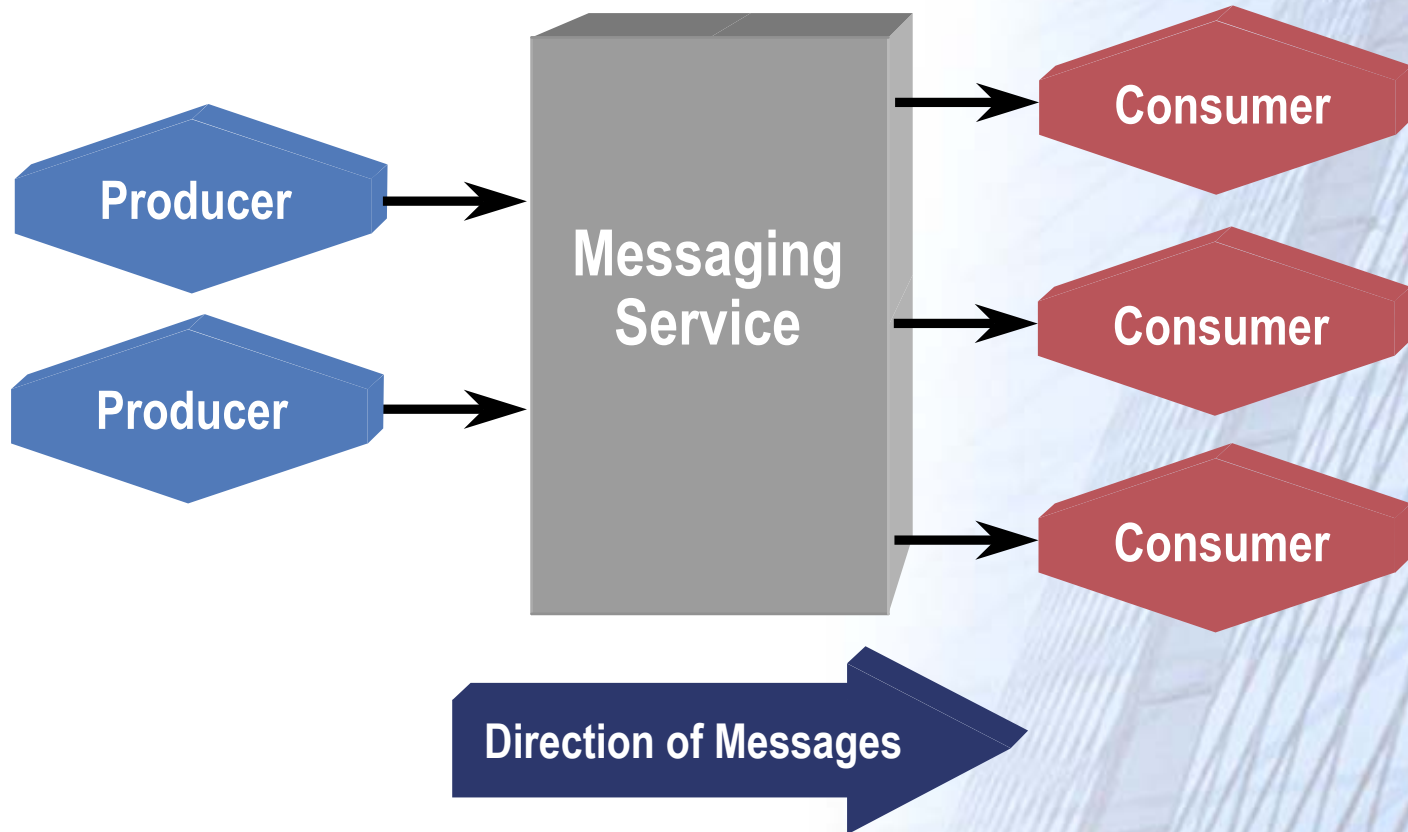
- **Why Messaging Systems?**
- **Concepts**
- **JMS specification**
  - Messaging Modes
  - Messages
  - Implementation
- **EJB 1.1 and JMS**
- **EJB 2.0 with Message Driven Beans**
- **Message Driven Beans**
- **Why Message Driven Beans?**

# Java's Messaging Service (JMS)

- **JMS is part of the J2EE specification**
- **JMS does not imply a particular implementation**
  - some implementations are built from ground-up
  - some implementations wrap existing systems
- **Asynchronous Communications in two Paradigms**
  - Publisher - Subscriber (Topics)
  - Queue (Point to Point)

# Messaging Service Concepts

Producers, consumers, and messaging services



# Outline

- Why Messaging Systems?
- Concepts
- **JMS specification**
  - Messaging Modes
  - Messages
  - Implementation
- EJB 1.1 and JMS
- EJB 2.0 with Message Driven Beans
- Message Driven Beans
- Why Message Driven Beans?

# Messaging Models

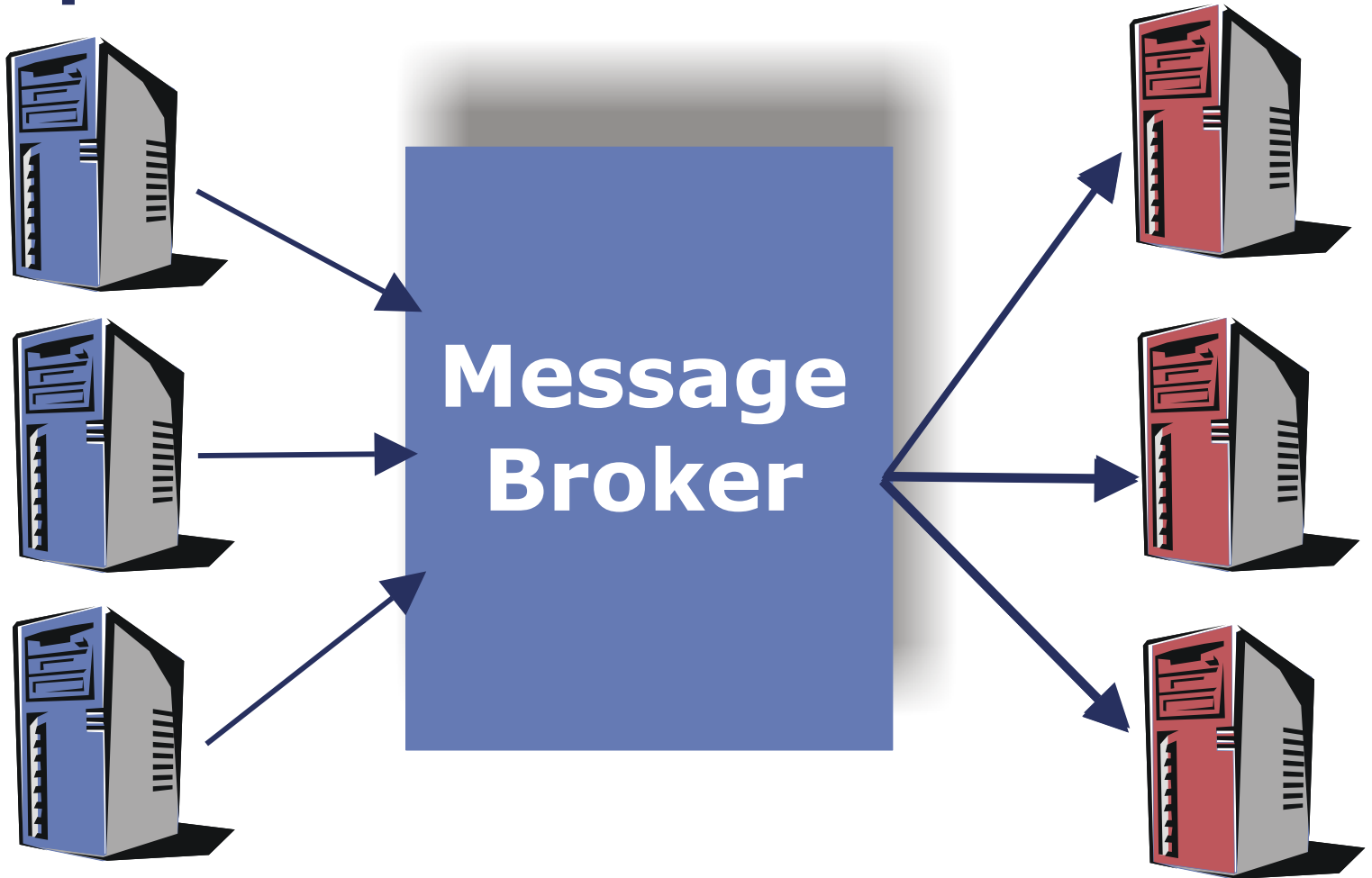
- **Publish and Subscribe**

- 0..n recipients
- Messages are passed via **Topics**
- **Publisher** and **Subscriber**

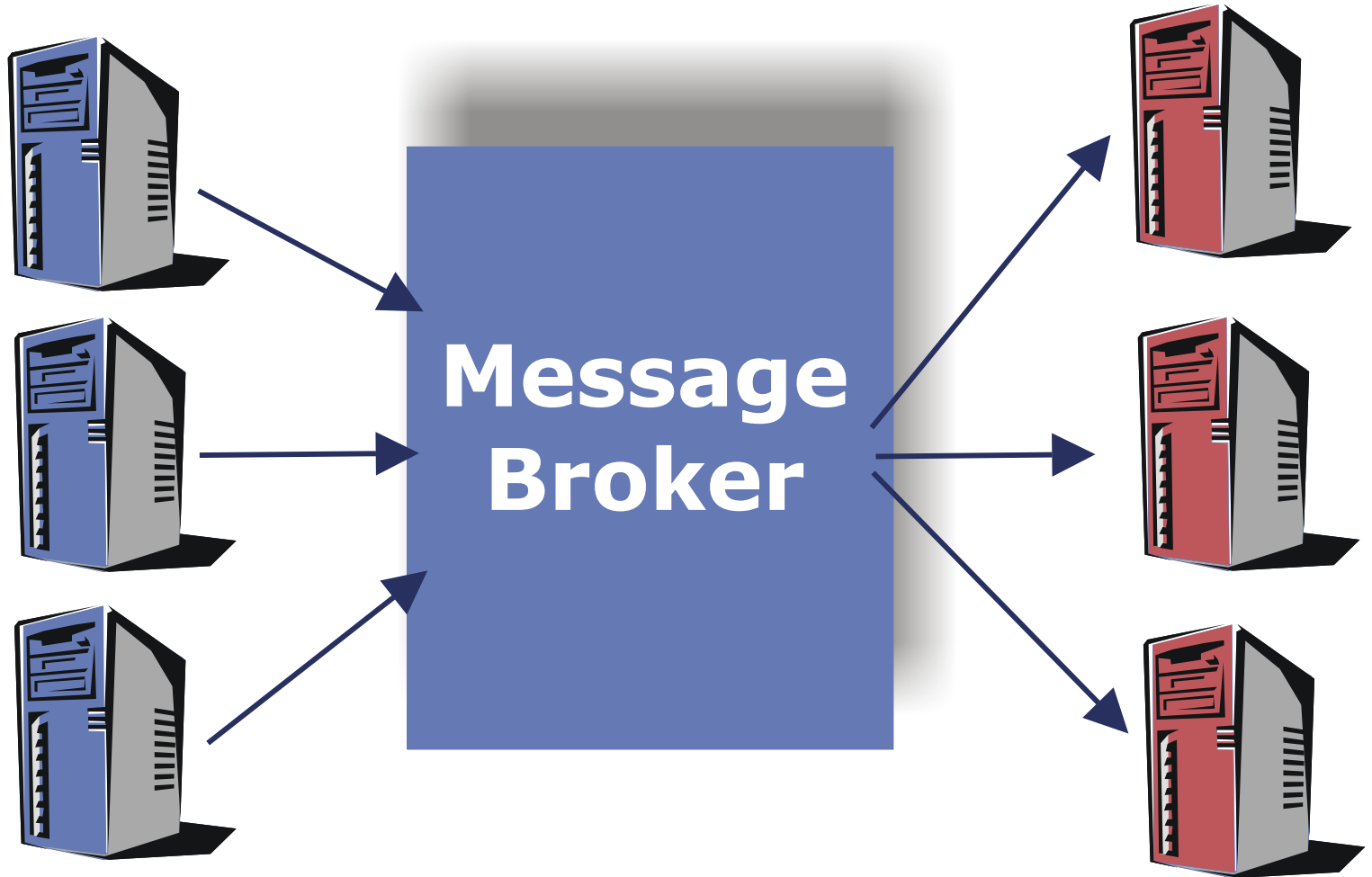
- **Point to Point**

- 1 recipient
- Messages are passed via **Queues**
- **Sender** and **Receiver**

# Topics



# Queue



# Outline

- Why Messaging Systems?
- Concepts
- **JMS specification**
  - Messaging Modes
  - **Messages**
  - Implementation
- EJB 1.1 and JMS
- EJB 2.0 with Message Driven Beans
- Message Driven Beans
- Why Message Driven Beans?

# Message Structure

- **JMS Messages have three parts**
  - **Message Header**
    - fixed set of fields defined by JMS
  - **Message Properties**
    - used for message filtering and selection
    - optional fields
    - defined by JMS specification, Application and JMS Provider (vendor)
  - **Message Body**
    - used to transport application specific information
    - more detail of message body types are on next slide

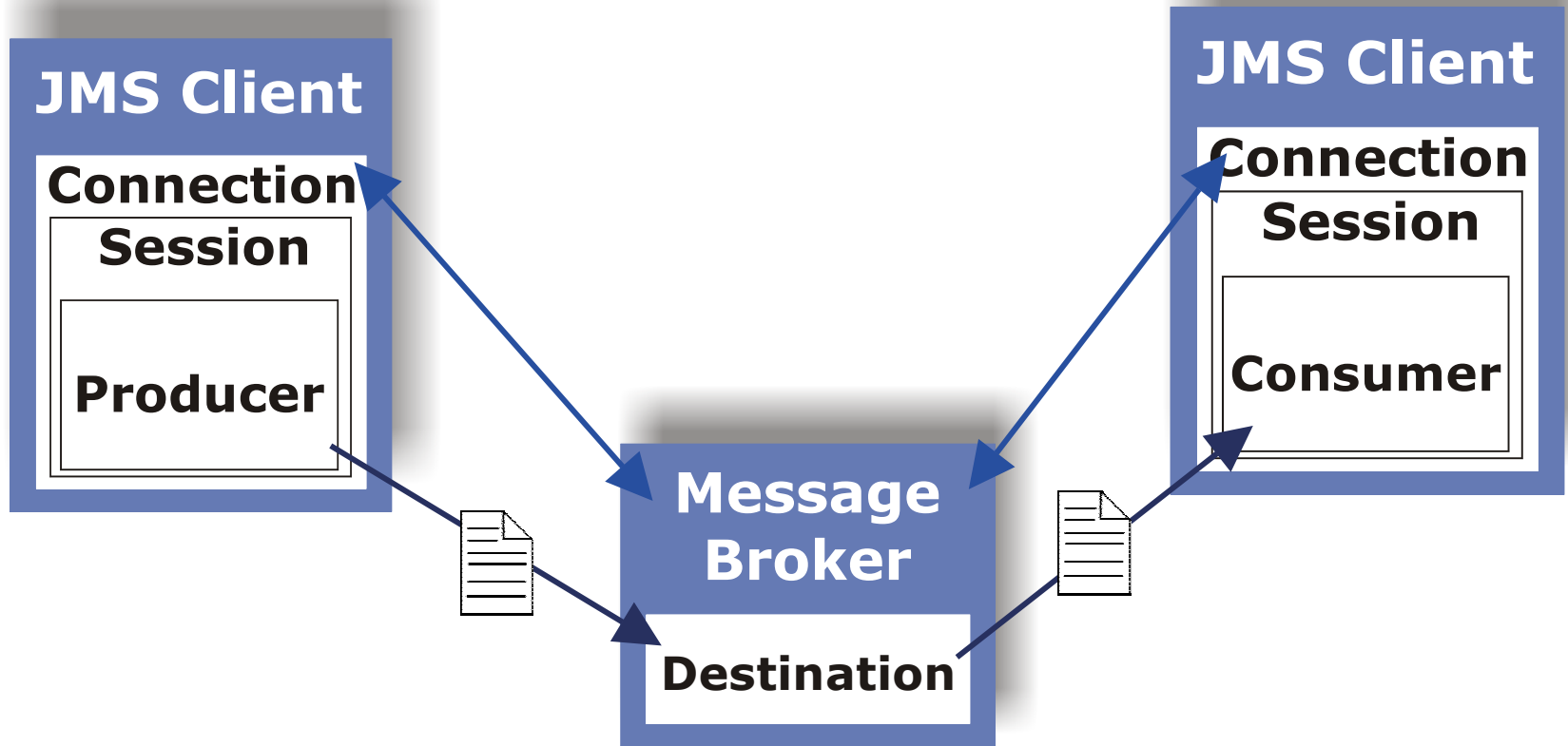
# Message Header

- **JMSDestination**
- **JMSDeliveryMode**
- **JMSMessageID**
- **JMSTimestamp**
- **JMSReplyTo**
- **JMSPriority**
- ...
- ...

# Message Types

- **JMS provides 5 types of message body**
  - **StreamMessage**
    - contains a stream of Java primitive types
  - **MapMessage**
    - contains a set of name-value pairs, where names are Strings and values are Java primitive types
  - **TextMessage**
    - contains one `java.lang.String`. XML can go here.
  - **ObjectMessage**
    - contains a `Serializable` Java object
  - **BytesMessage**
    - contains a stream of uninterpreted bytes

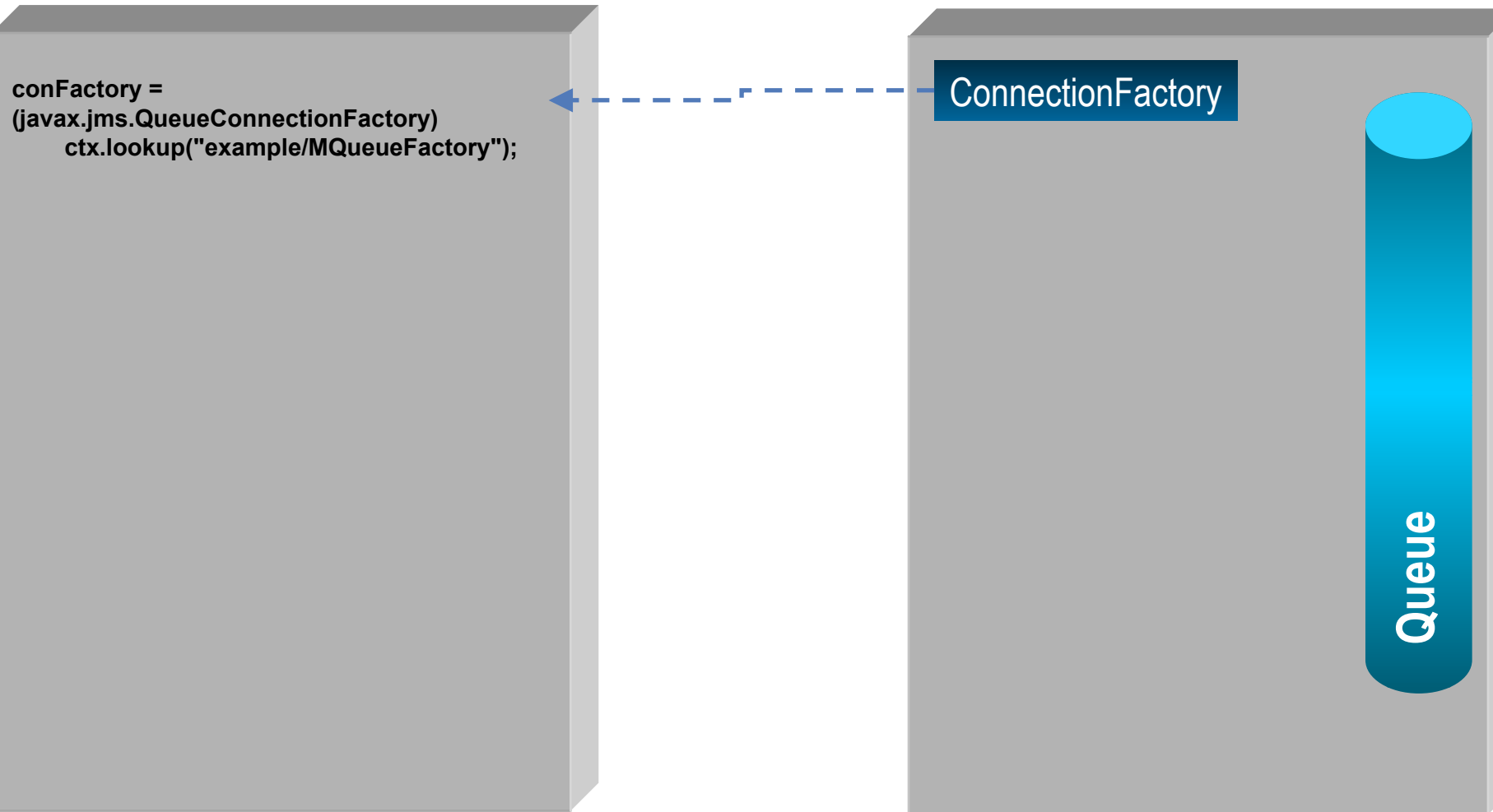
# JMS Components



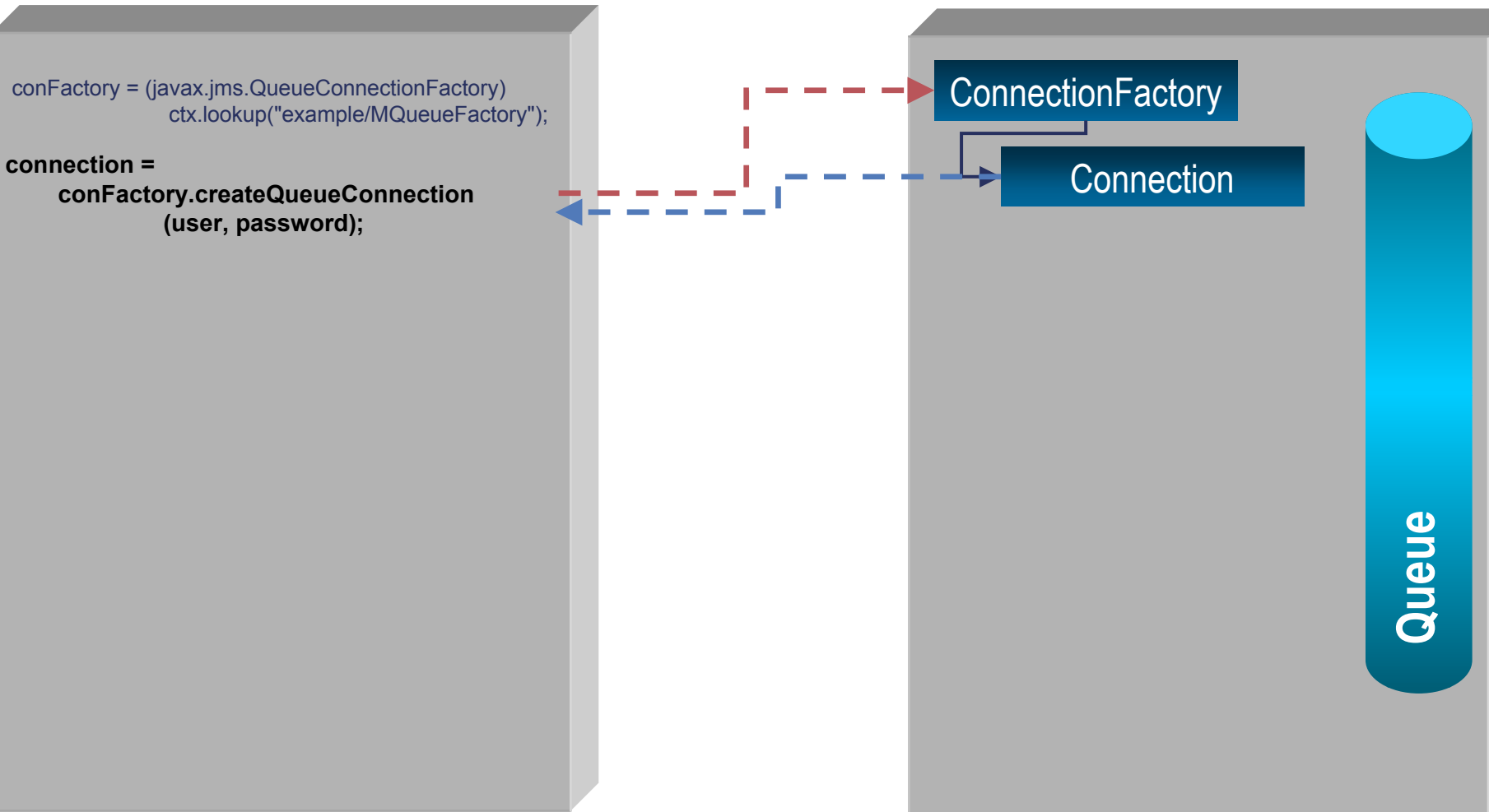
# Outline

- Why Messaging Systems?
- Concepts
- **JMS specification**
  - Messaging Modes
  - Messages
  - **Implementation**
- EJB 1.1 and JMS
- EJB 2.0 with Message Driven Beans
- Message Driven Beans
- Why Message Driven Beans?

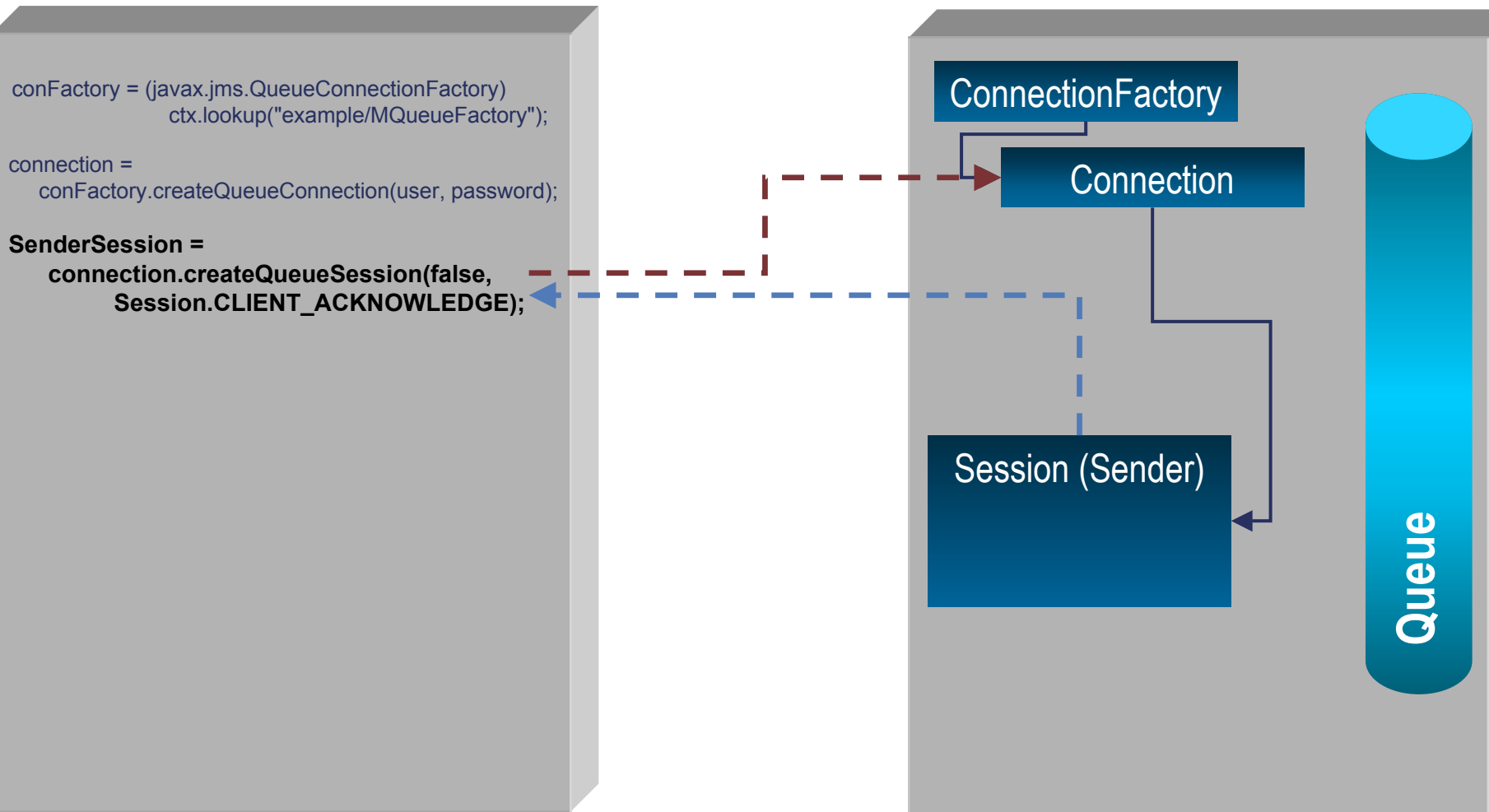
# JMS Client Bootstrapping



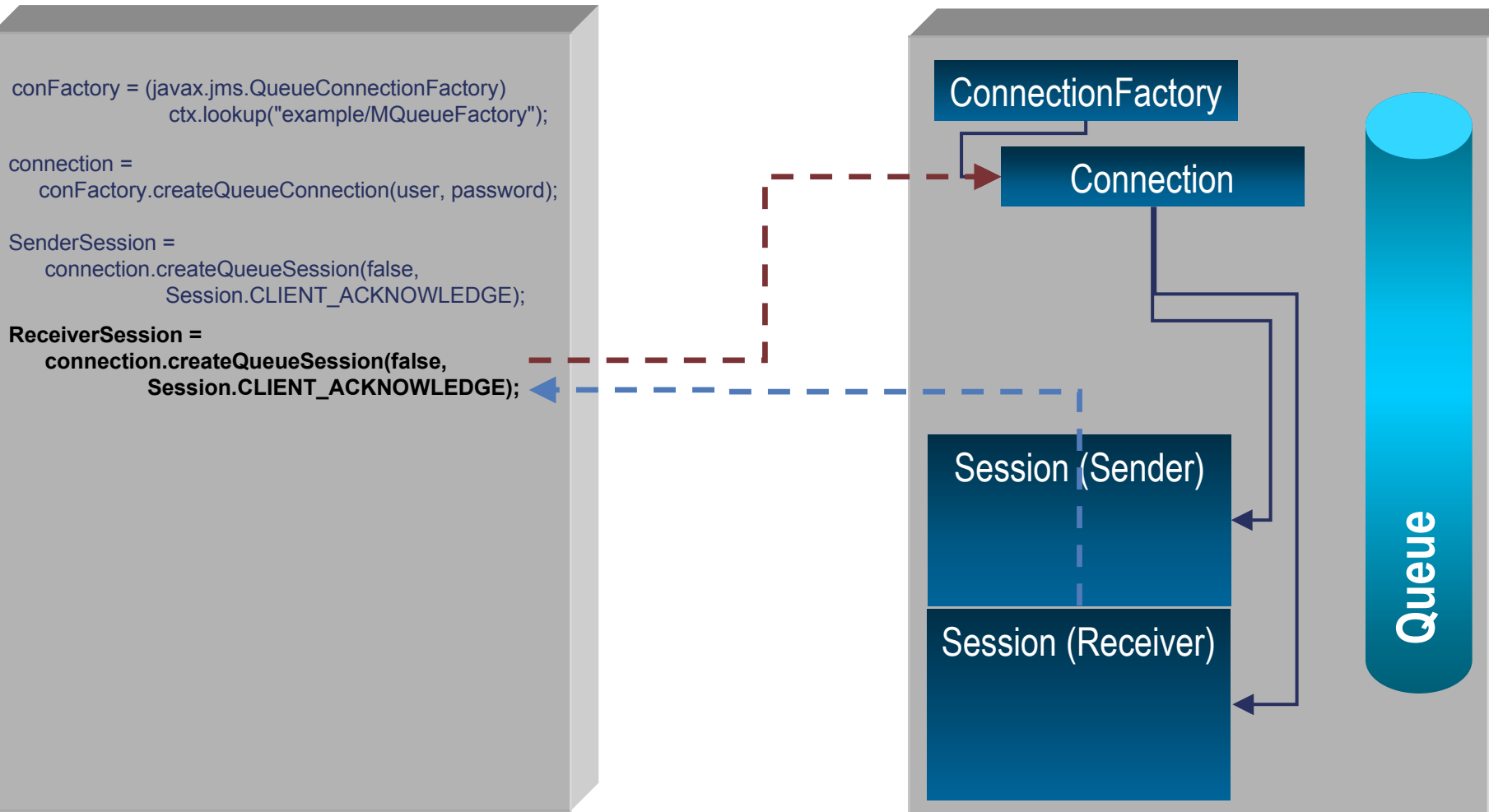
# JMS Client Bootstrapping



# JMS Client Bootstrapping



# JMS Client Bootstrapping



# JMS Client Bootstrapping

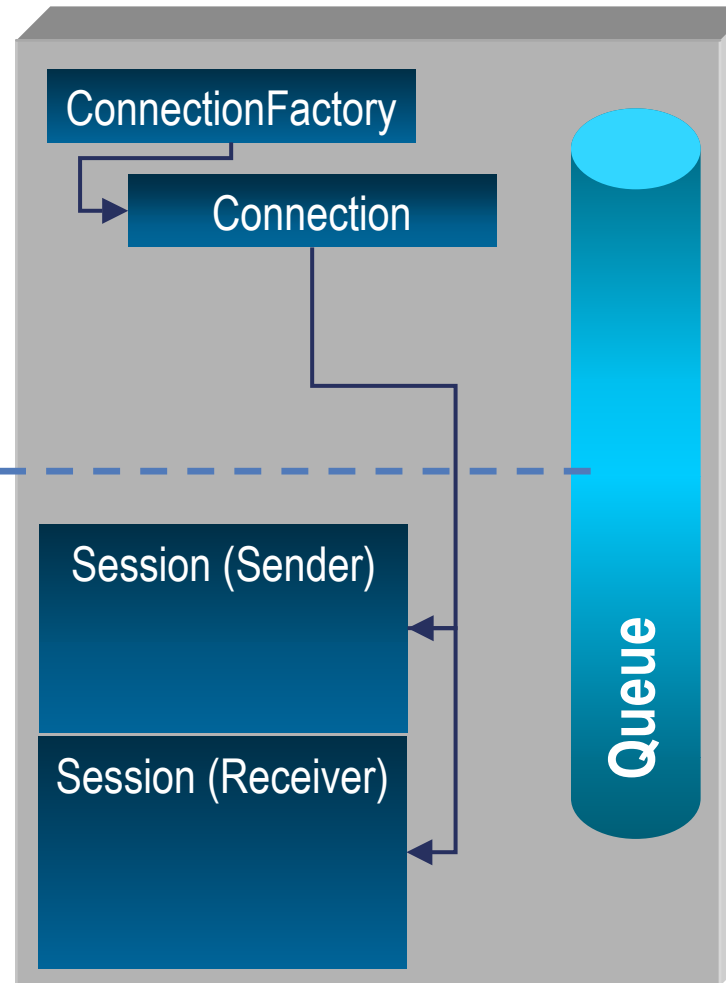
```
conFactory = (javax.jms.QueueConnectionFactory)
    ctx.lookup("example/MQueueFactory");

connection =
    conFactory.createQueueConnection(user, password);

SenderSession =
    connection.createQueueSession(false,
        Session.CLIENT_ACKNOWLEDGE);

ReceiverSession =
    connection.createQueueSession(false,
        Session.CLIENT_ACKNOWLEDGE);

TickerQueue = (javax.jms.Queue)
    ctx.lookup(QueueName);
```



# JMS Client Bootstrapping

```
conFactory = (javax.jms.QueueConnectionFactory)
    ctx.lookup("example/MQueueFactory");

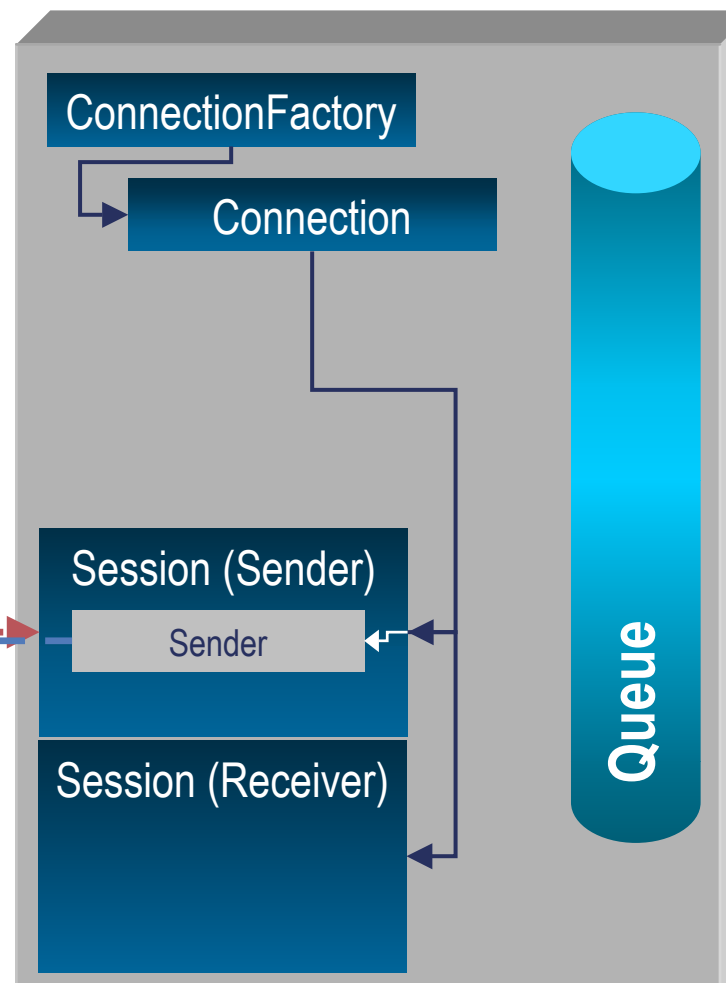
connection =
    conFactory.createQueueConnection(user, password);

SenderSession =
    connection.createQueueSession(false,
        Session.CLIENT_ACKNOWLEDGE);

ReceiverSession =
    connection.createQueueSession(false,
        Session.CLIENT_ACKNOWLEDGE);

TickerQueue = (javax.jms.Queue)
    ctx.lookup(QueueName);

queueSender =
    SenderSession.createSender(TickerQueue);
```





# JMS Client Bootstrapping

```
conFactory = (javax.jms.QueueConnectionFactory)
    ctx.lookup("example/MQueueFactory");

connection =
    conFactory.createQueueConnection(user, password);

SenderSession =
    connection.createQueueSession(false,
        Session.CLIENT_ACKNOWLEDGE);

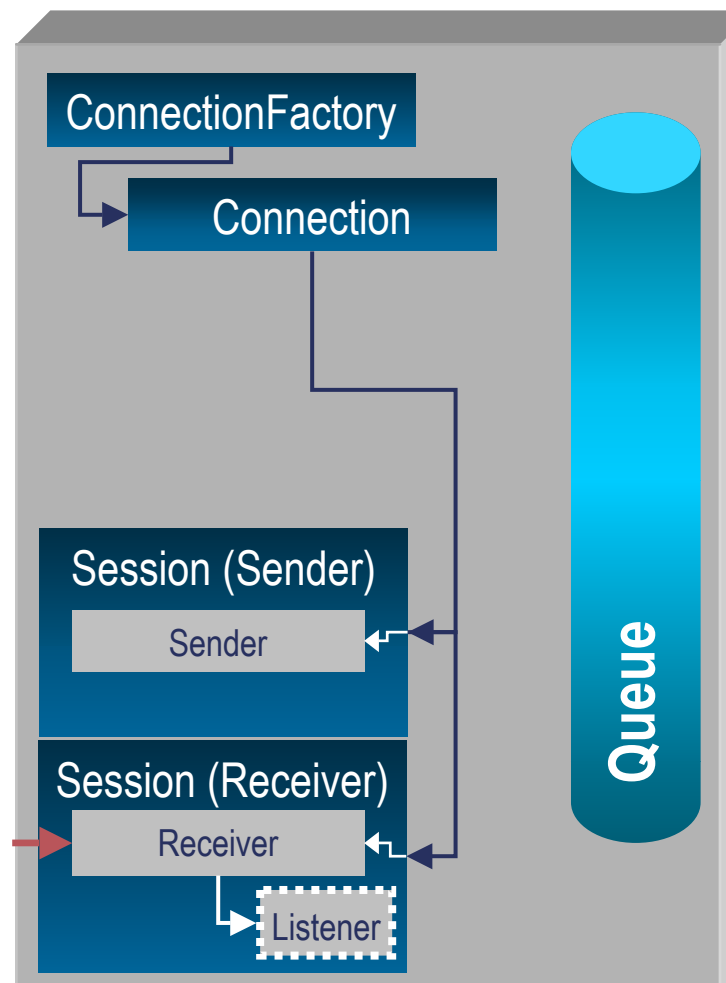
ReceiverSession =
    connection.createQueueSession(false,
        Session.CLIENT_ACKNOWLEDGE);

TickerQueue = (javax.jms.Queue)
    ctx.lookup(QueueName);

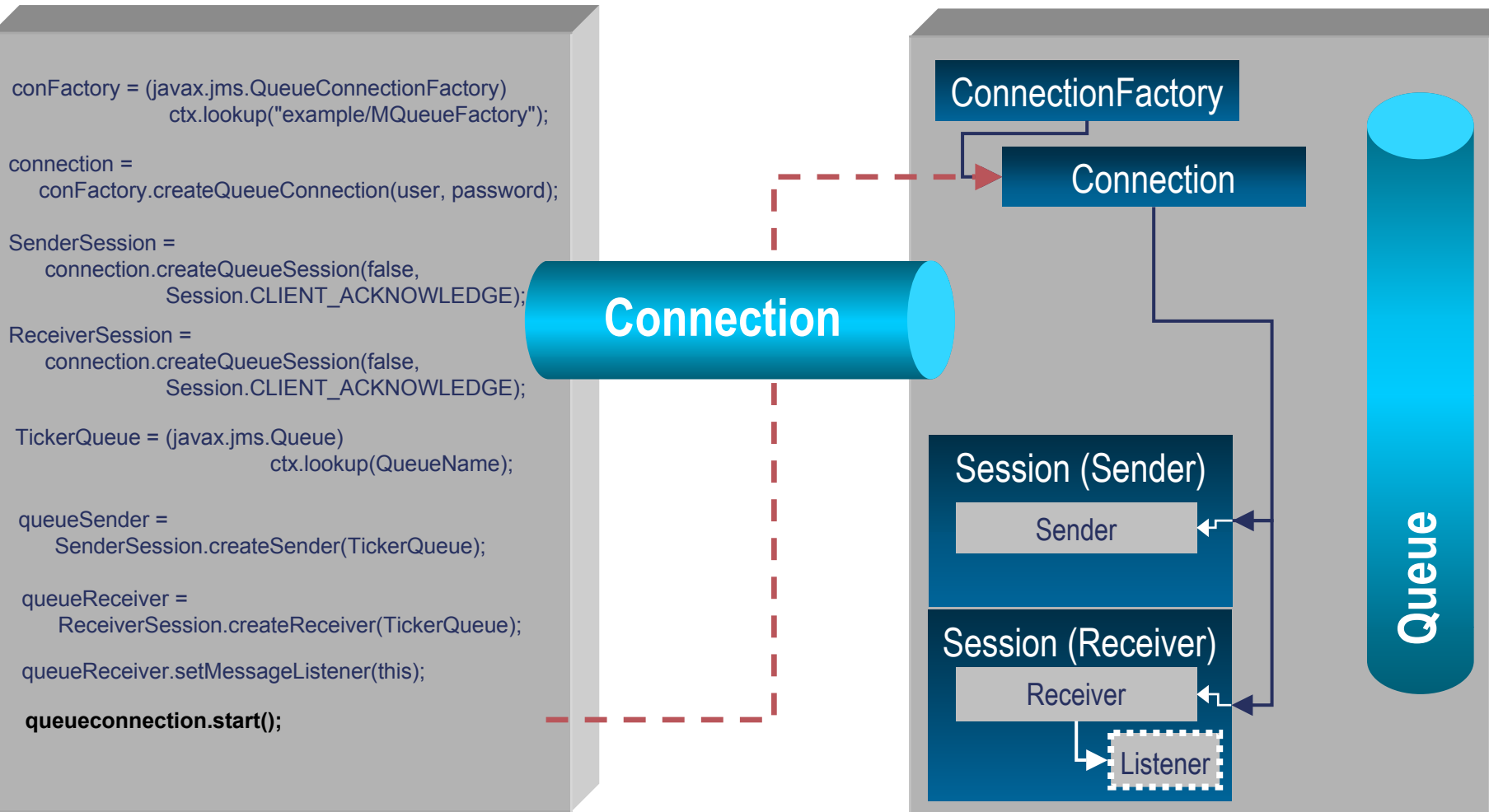
queueSender =
    SenderSession.createSender(TickerQueue);

queueReceiver =
    ReceiverSession.createReceiver(TickerQueue);

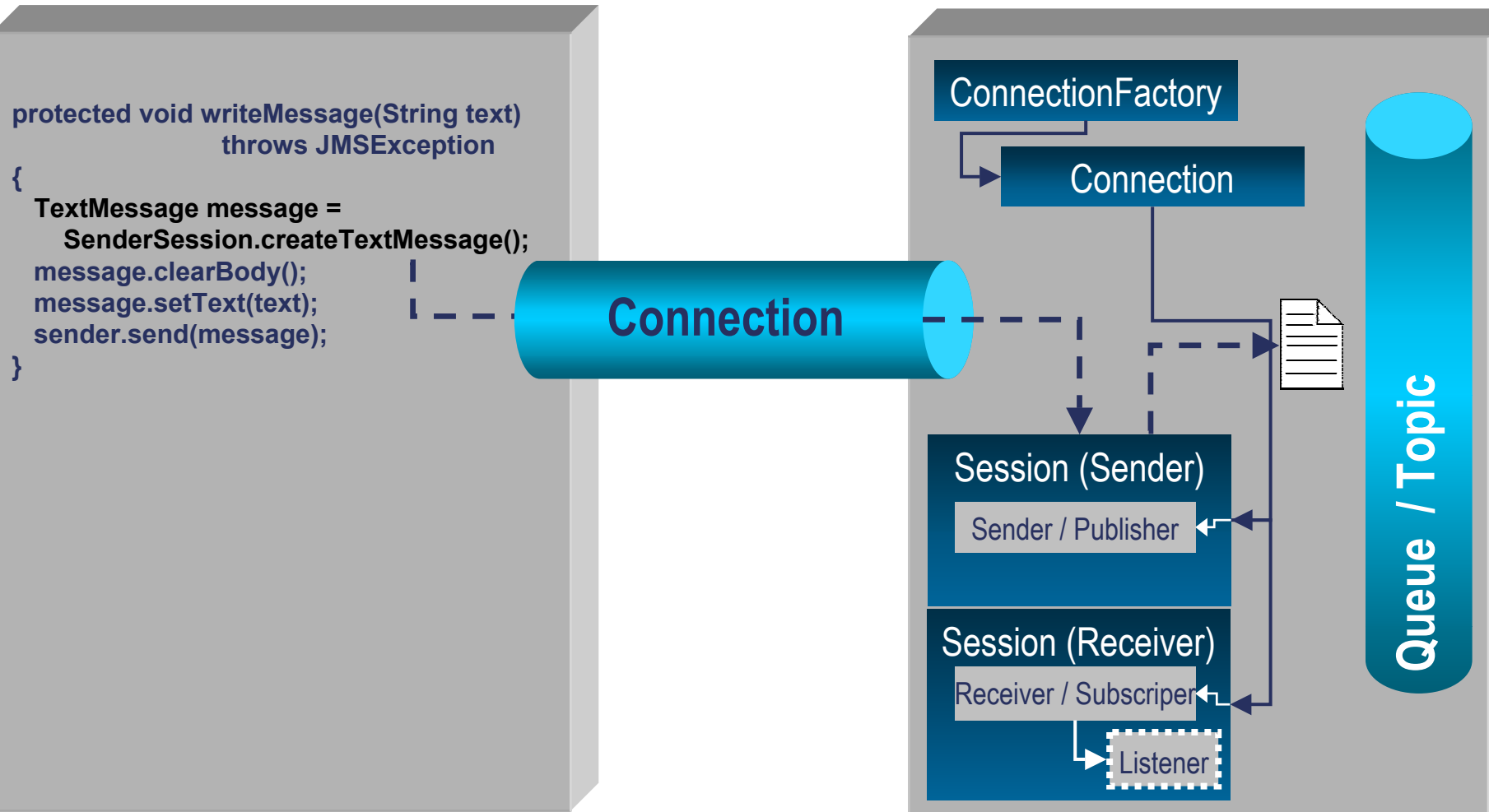
queueReceiver.setMessageListener(this);
```



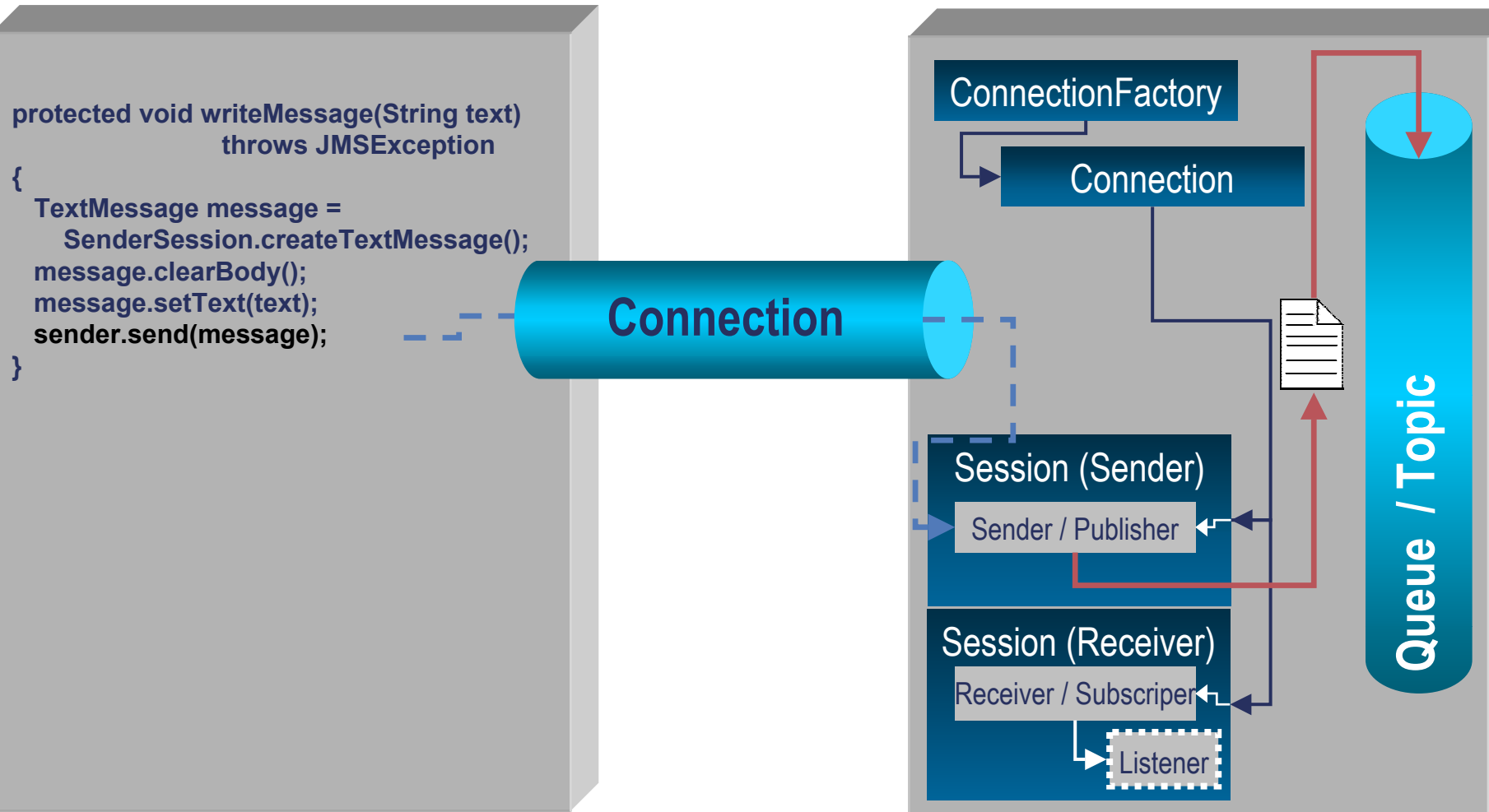
# JMS Client Bootstrapping



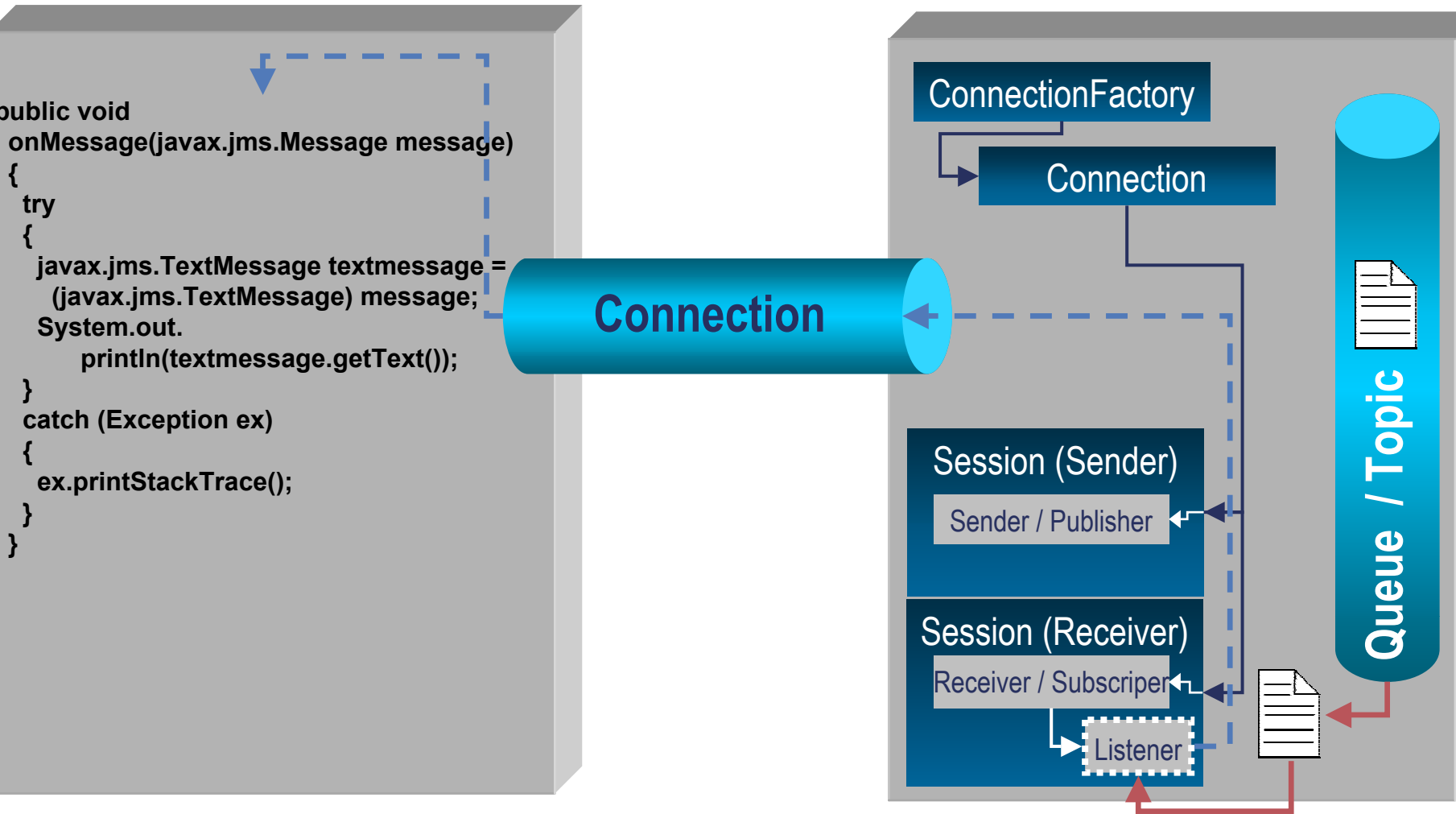
# Send Message



# Send Message



# Receive Message



# Outline

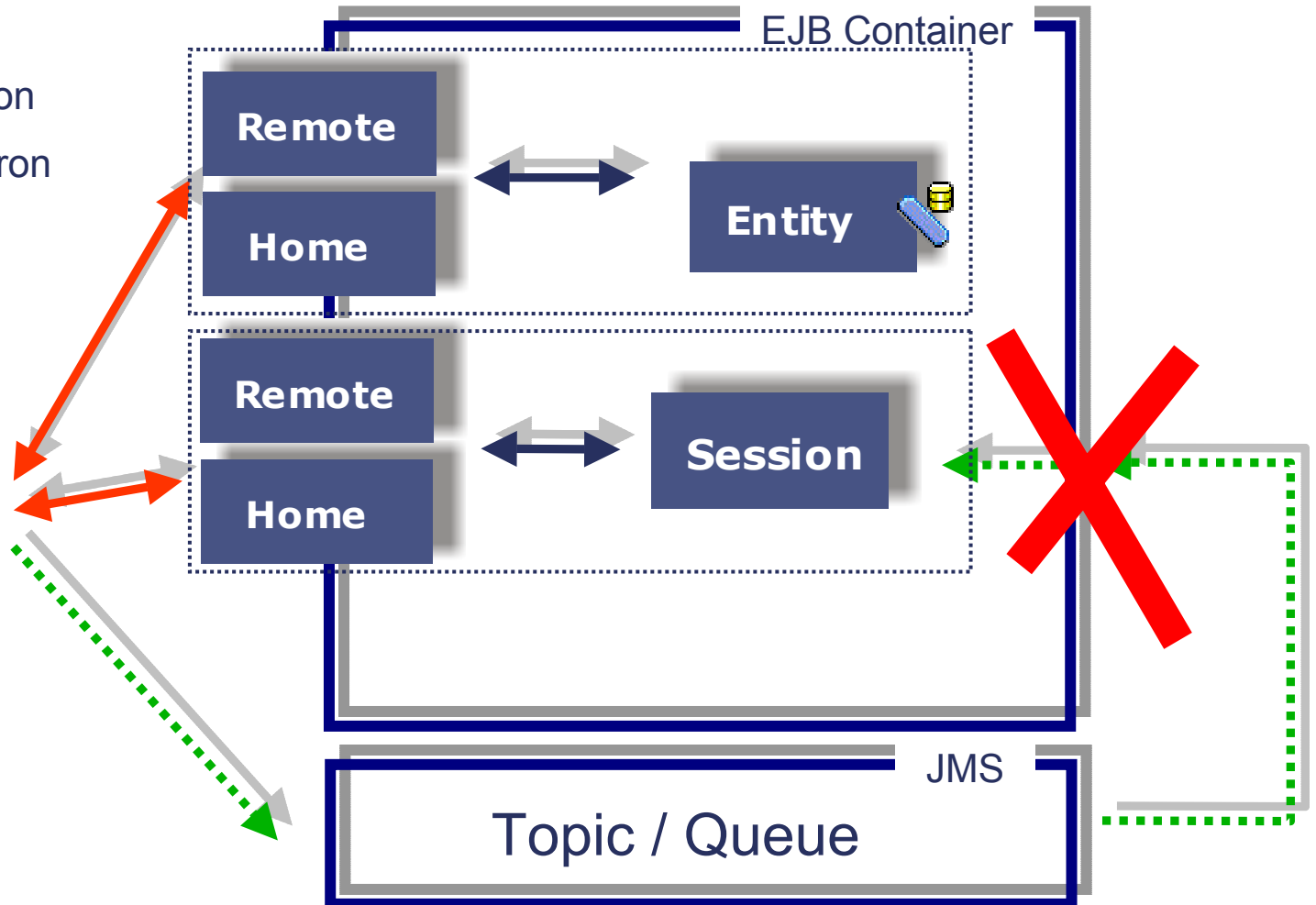
- **Why Messaging Systems?**
- **Concepts**
- **JMS specification**
  - Messaging Modes
  - Messages
  - Implementation
- **EJB 1.1 and JMS**
- **EJB 2.0 with Message Driven Beans**
- **Message Driven Beans**
- **Why Message Driven Beans?**

# Enterprise Java Beans (Spec. 1.1)

client view



Client



# EJB couldn't be a JMS Listener (EJB1.1)

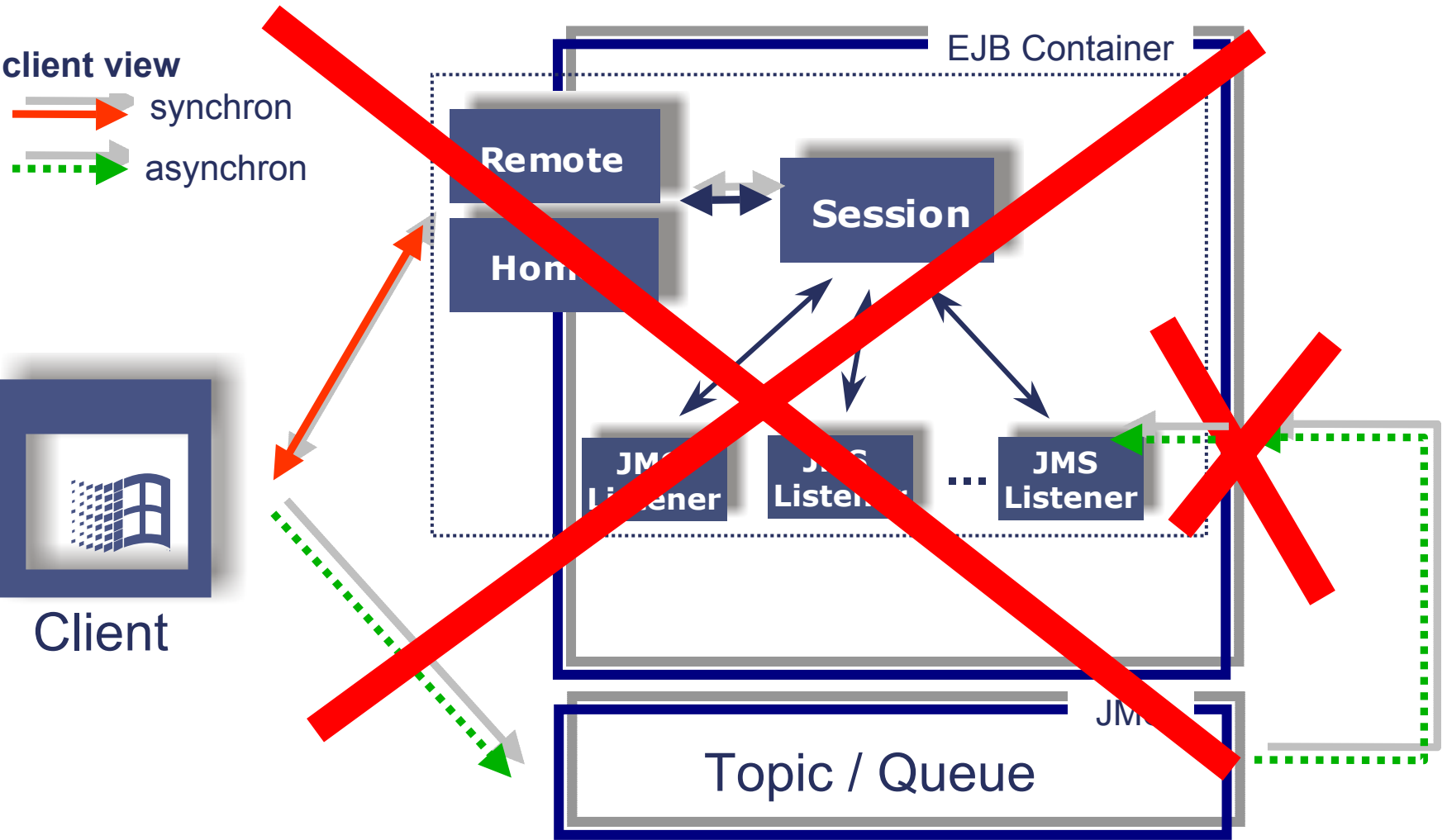
- **JMS Listener has to implement**

- `javax.jms.MessageListener` (**Interface**)
- `onMessage(javax.jms.Message m)` (**Method**)

- **An EJB client communicate with an EJB over Home/Remote objects which are only defined as interfaces**

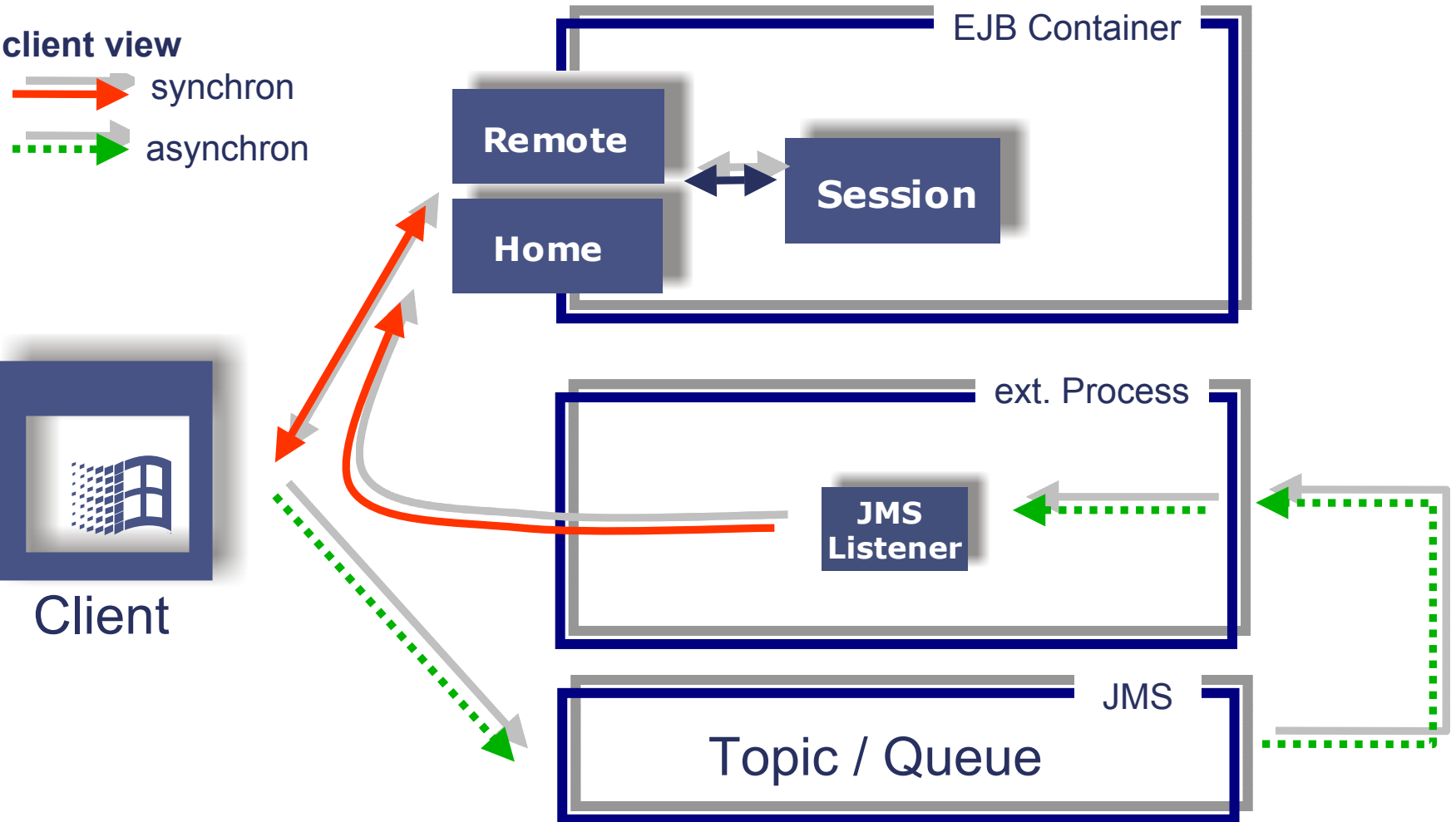
- **Problem:** JMS mustn't call `onMessage()` of the EJB class.

# Be careful (Spec. 1.1)



# JMS Workaround (Spec. 1.1)

client view



# JMS with EJB (EJB1.1)

- EJBs can be a Sender/Publisher

- EJBs cannot be a JMS Listener

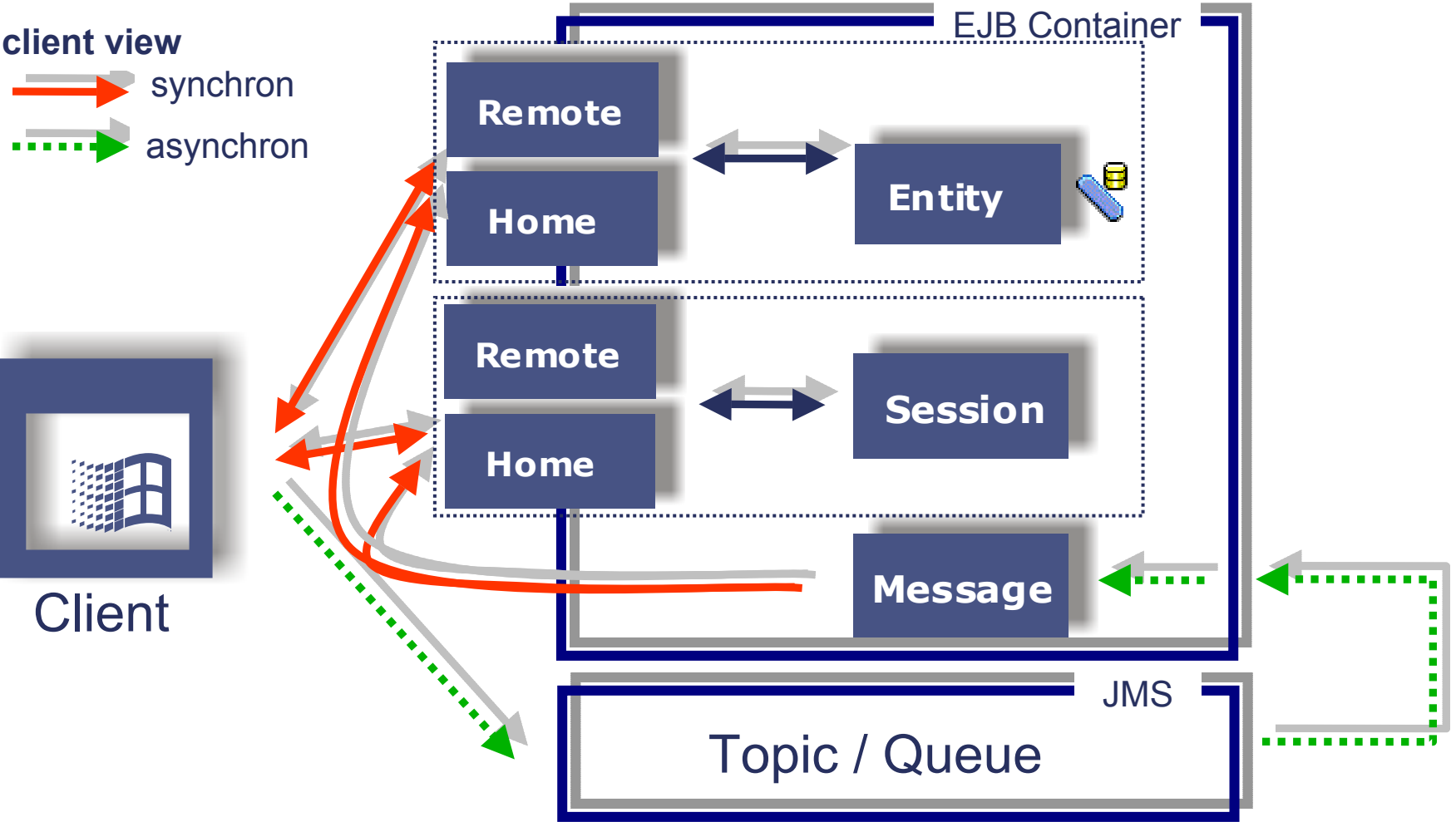
EJBs only receive messages (Polling) with:

- Message receive();
- Message receive(long timeout);
- Message receiveNoWait();

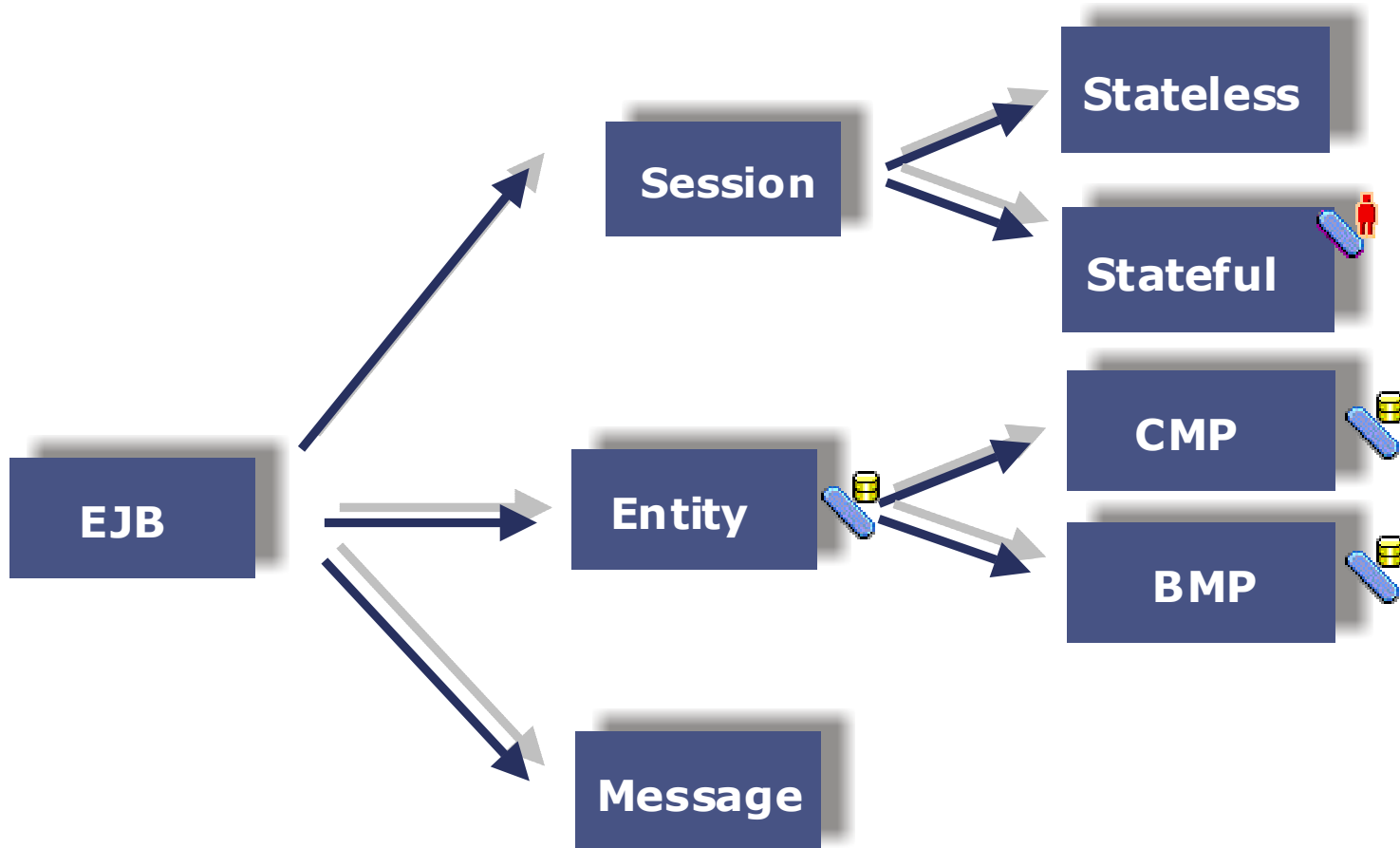
# Outline

- Why Messaging Systems?
- Concepts
- JMS specification
  - Messaging Modes
  - Messages
  - Implementation
- EJB 1.1 and JMS
- **EJB 2.0 with Message Driven Beans**
- Message Driven Beans
- Why Message Driven Beans?

# Enterprise Java Beans (Spec. 2.0)



# Enterprise Java Beans (Spec. 2.0)



# Outline

- Why Messaging Systems?
- Concepts
- JMS specification
  - Messaging Modes
  - Messages
  - Implementation
- EJB 1.1 and JMS
- EJB 2.0 with Message Driven Beans
- **Message Driven Beans**
- Why Message Driven Beans?

# Message Driven Bean (MDB) (1)

- **A message-driven bean:**

- is an asynchronous message consumer.
- is invoked by the container as a result of the arrival of a JMS message
- has neither a home nor a remote interface.
- are anonymous. They have no client-visible identity.
- is an instance of a message-driven bean class

# Message Driven Bean (MDB) (2)

## ■ A message-driven bean:

- Executes on receipt of a single client message.
- May update shared data in an underlying database.
- Does not represent directly shared data in the database, although it may access and update such data.
- Is relatively short-lived.
- Is stateless.
- Is removed when the EJB Container crashes.

# MDB - Client View

- To a client, a **message-driven bean is a JMS message consumer** that implements some business logic running on the server.
- A **client accesses a message-driven bean through JMS** by sending messages to the JMS Destination (Queue or Topic) for which the message-driven bean class is the MessageListener.
- **Message-driven bean instances have no conversational state.** This means that all bean instances are equivalent when they are not involved in servicing a client message.

# Implementation of a MDB

## ■ The class must:

- implement, the `javax.ejb.MessageDrivenBean` and the `javax.jms.MessageListener` **interface**
- be defined as `public`
- must not be `final`
- must not be `abstract`
- must implement the `ejbCreate()` method
- must not define the `finalize()` method

# Implementation example

```
import javax.naming.*;
import javax.jms.*;
import javax.ejb.*;

public class HelloBean implements MessageDrivenBean,
                                   MessageListener {
    private MessageDrivenContext _mdc;
    ...
    public void onMessage(Message msg) {
        {
            System.out.println("HelloBean.onMessage() called.");
        }
    }
}
```

# Deployment Descriptor Editor

The screenshot shows the JBuilder - MyMDBBean Deployment Descriptor Editor. The window title is "JBuilder - MyMDBBean". The menu bar includes File, Edit, Search, View, Project, Run, Team, Wizards, Tools, Window, and Help. The toolbar contains various icons for file operations and development actions. The left pane shows a project tree with the following structure:

- untitled3.jpx
  - untitled3
  - Additional Settings
  - Untitled1
    - SLSB2
    - Account
    - MyMDBBean**
    - Container Transact
  - JDBC 1 Datasources
  - Security Roles
  - Untitled1.jar
- client.java
- jndi-definitions.xml

The right pane displays the configuration for the selected **Message-Driven Bean 'MyMDBBean'**. The fields are:

- Bean name: MyMDBBean
- Bean class: untitled3.MyMDBBeanBean
- Description: (empty text area)
- Icons:
  - Small icon (16X16): (empty text area)
  - Large icon (32X32): (empty text area)

At the bottom, there are tabs for configuration sections: General, Message-Driven Bean (selected), Environment, EJB Local References, EJB References, Resource References, Resource Env Refs, Properties, and Security Identity. The status bar at the bottom left indicates "Reverted 0 modified files."

# Deployment Descriptor Editor

**JBuilder - MyMDBBean**

File Edit Search View Project Run Team Wizards Tools Window Help

untitled3.jpx  
untitled3  
Additional Settings  
Untitled1  
SLSB2  
Account  
**MyMDBBean**  
Container Transact  
JDBC 1 Datasources  
Security Roles  
Untitled1.jar  
client.java  
jndi-definitions.xml

### Message-Driven Bean 'MyMDBBean'

Transaction type:  
Container

Message selector:  
[Empty]

Destination

Connection factory name:  
serial://jms/tcf

Destination name:  
serial://jms/t

Destination type:  
javax.jms.Topic

Subscription durability:  
Durable

Pool

Initial pool size:  
0

Maximum pool size:  
0

Wait timeout:  
0

General Message-Driven Bean Environment EJB Local References EJB References  
Resource References Resource Env Refs Properties Security Identity

Reverted 0 modified files.

# MDB and Transactions

- Can be transaction-aware.
- Message Driven Beans are only allowed to have to following demarcations:
  - **Required**
  - **Not\_Supported**

# MDB and Acknowledgment

- MDB-Container send an acknowledgment to the JMS provider.
- Acknowledge modes:
  - **Auto-acknowledge**
  - **Dups-ok-acknowledge**

# Transactions and Acknowledgment in DD

- Transaction types:

- **Required**

- Acknowledgment mode is ignored

- Transaction commit -> message is acknowledged

- Transaction rollback -> message is **not** acknowledged

- **Not\_Supported**

- Acknowledgment mode is *important*

# Outline

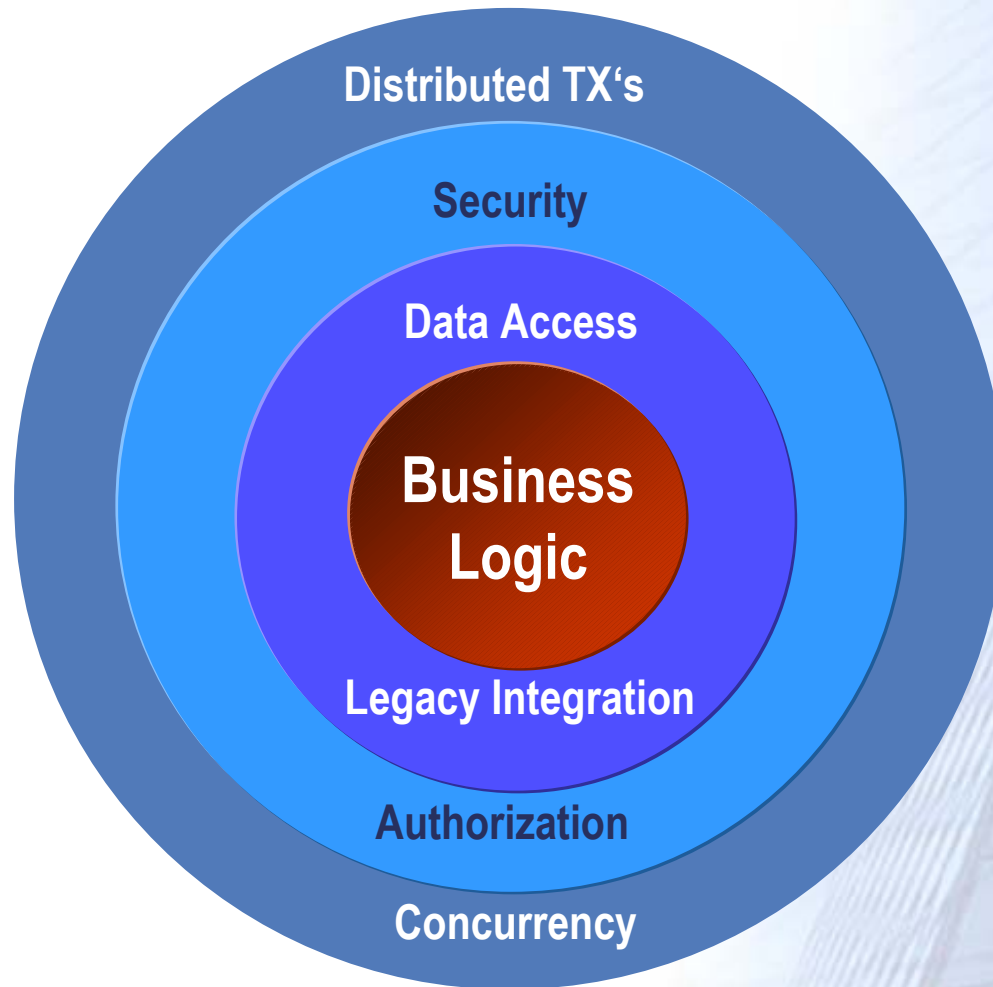
- **Why Messaging Systems?**
- **Concepts**
- **JMS specification**
  - Messaging Modes
  - Messages
  - Implementation
- **EJB 1.1 and JMS**
- **EJB 2.0 with Message Driven Beans**
- **Message Driven Beans**
- **Why Message Driven Beans?**

# Why Message Driven Beans? (1)

- **Message Driven Beans combines the advantages of**
  - J2EE architecture and
  - Java Messaging Systems

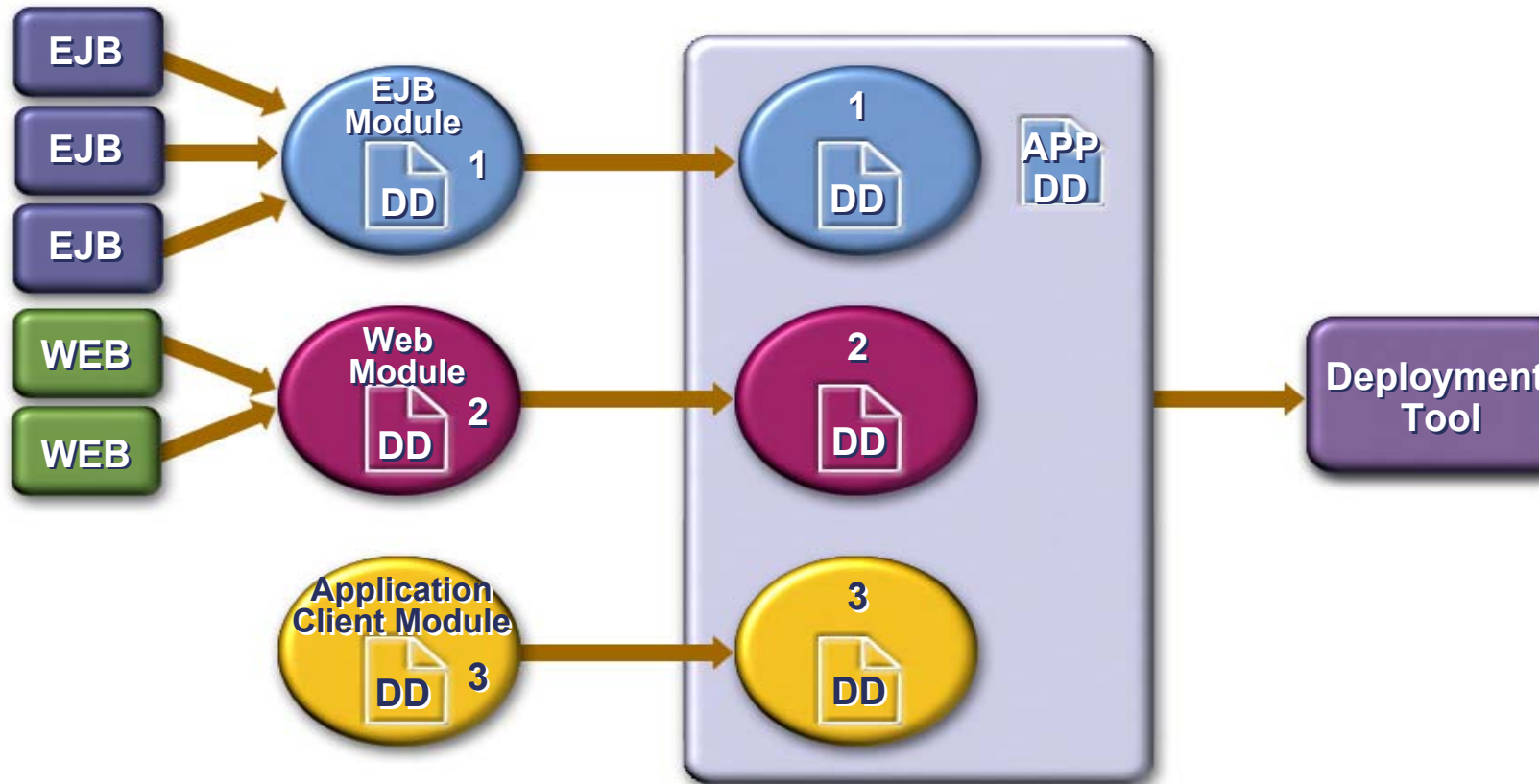
# Why Message Driven Beans? (2)

e.g. the  
EJB  
concept



# Why Message Driven Beans? (3)

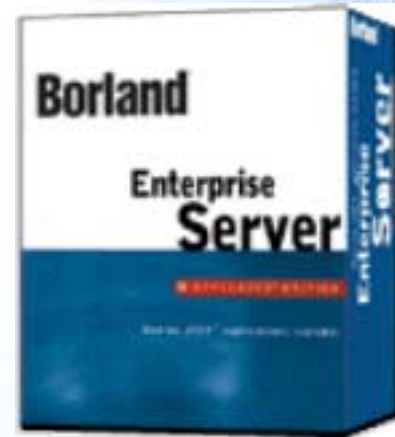
e.g. Application Packaging



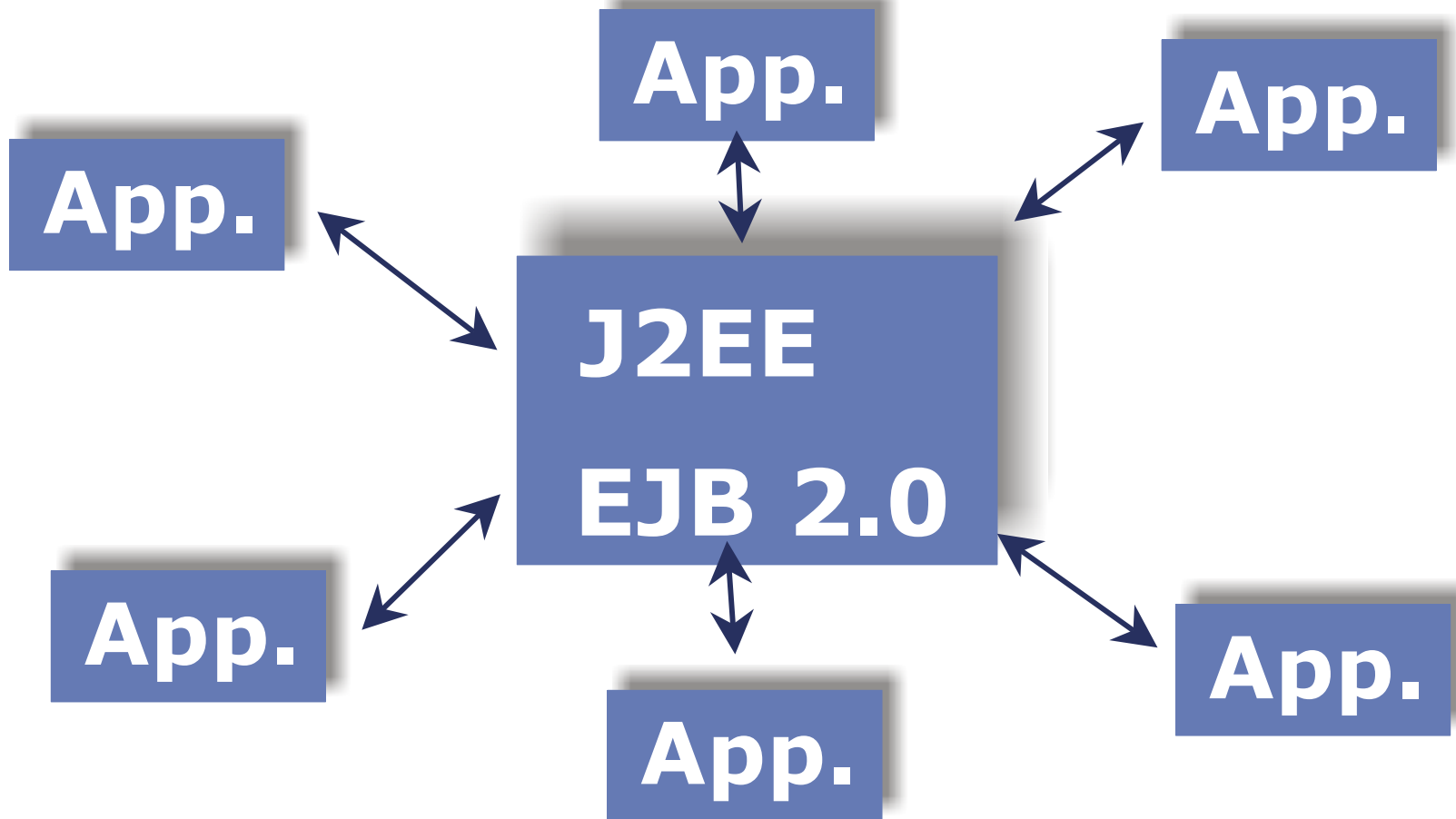
# Why Message Driven Beans? (4)

**J2EE  
Application Server**

**Java  
Messaging Service**



# Hub & Spoke now with J2EE-Server



SCALABILITY SECURITY

**Thank You**

EJB™

RELIABILITY

Web Services

J2EE

Borland