Module 10

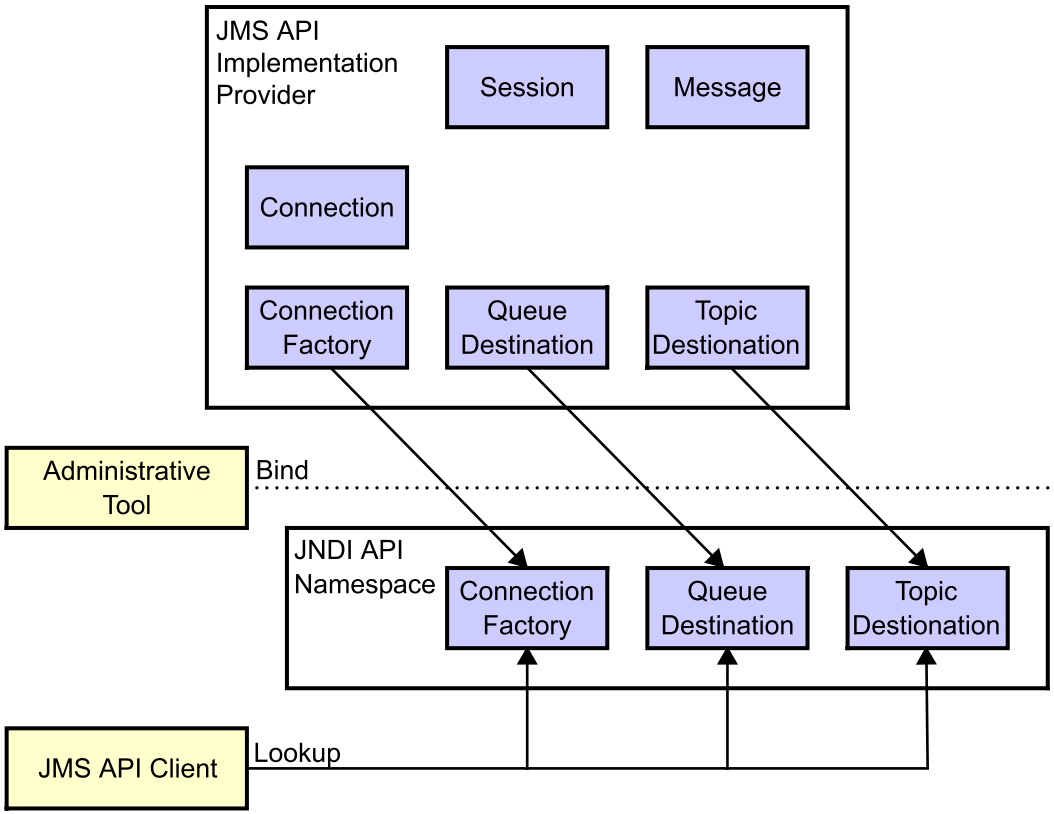
Developing Java EE Applications Using Messaging

Objectives

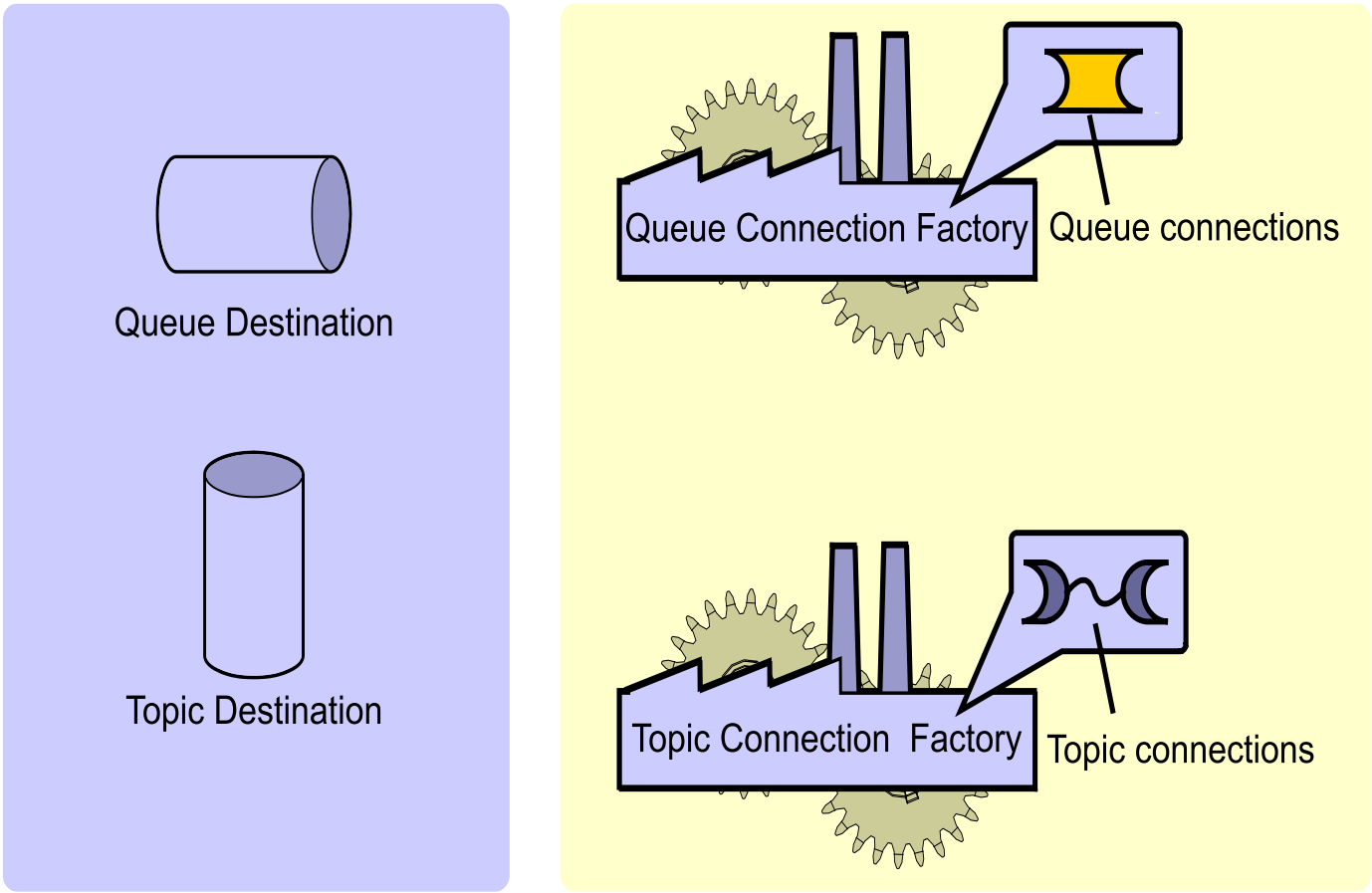
Upon completion of this module, you should be able to:

- Describe JMS API technology
- Write a message producer
- Write an asynchronous message consumer
- Write a synchronous message consumer
- List the capabilities and limitations of EJB components as messaging clients

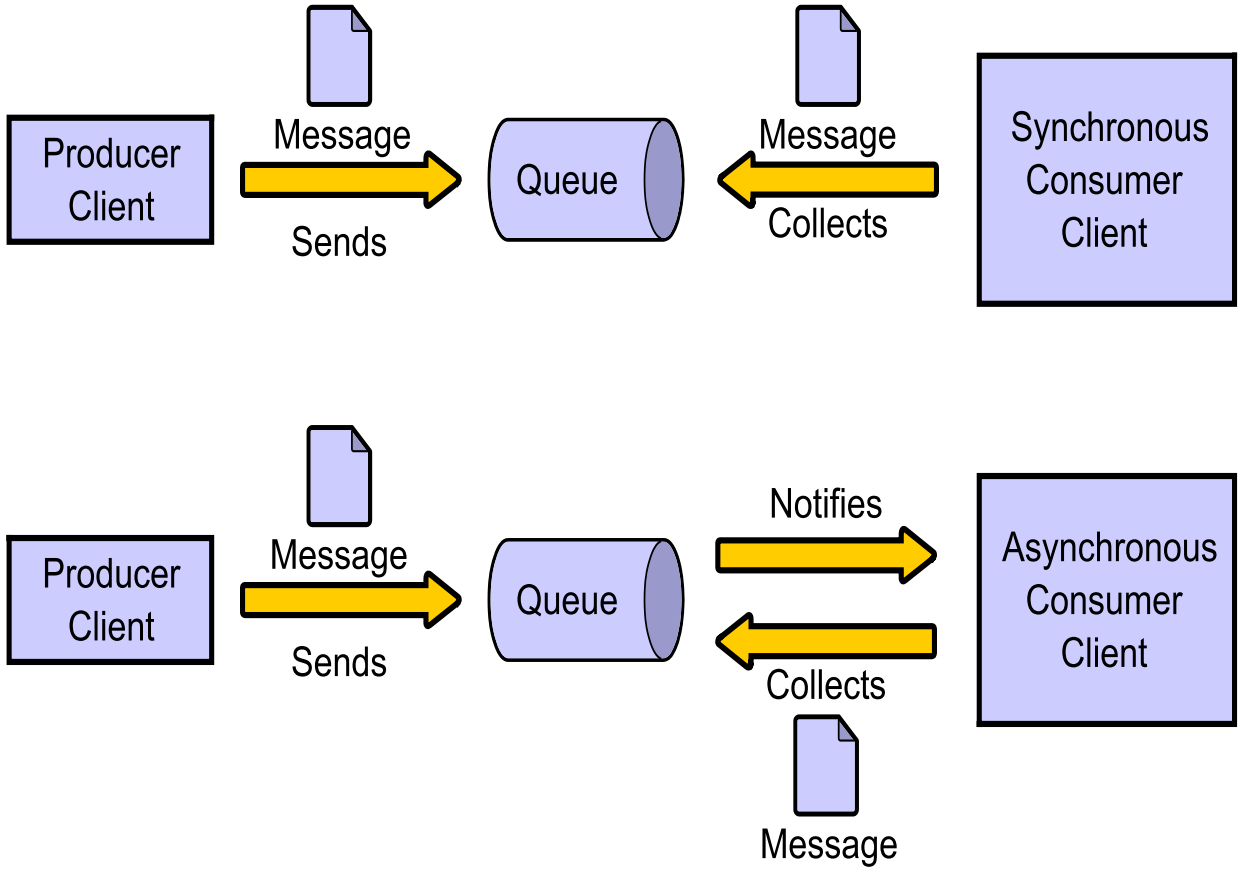
Messaging System Participants



Administered Objects

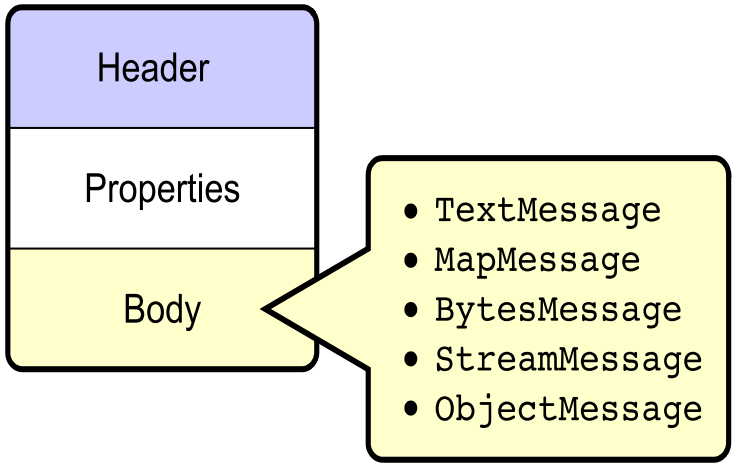


Messaging Clients



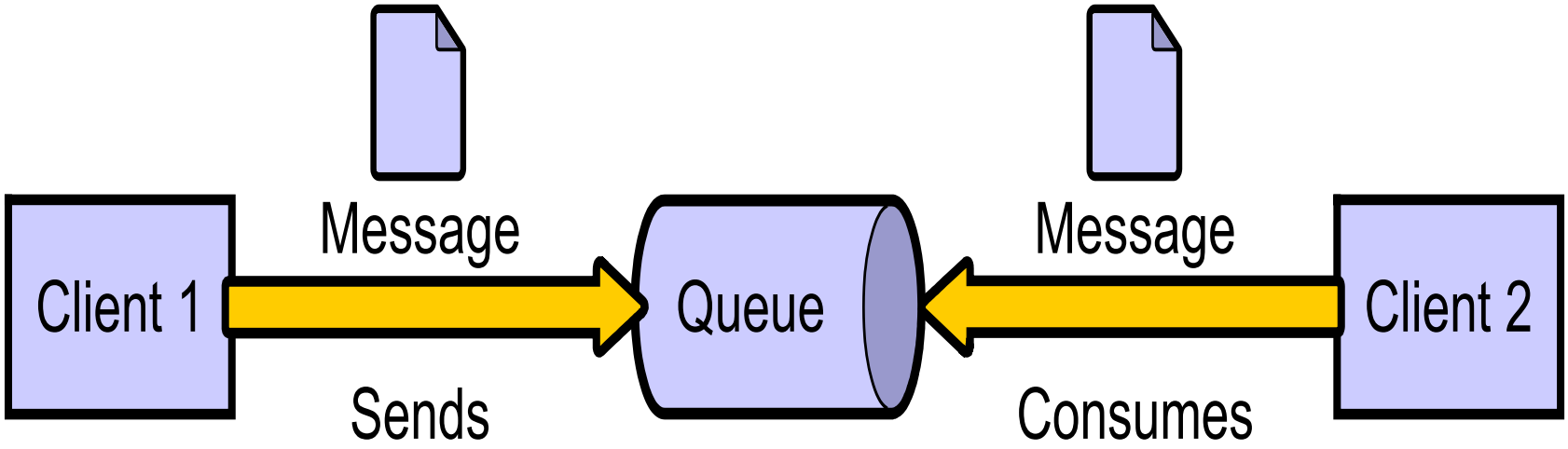
The synchronous client blocks until a message arrives.

Messages

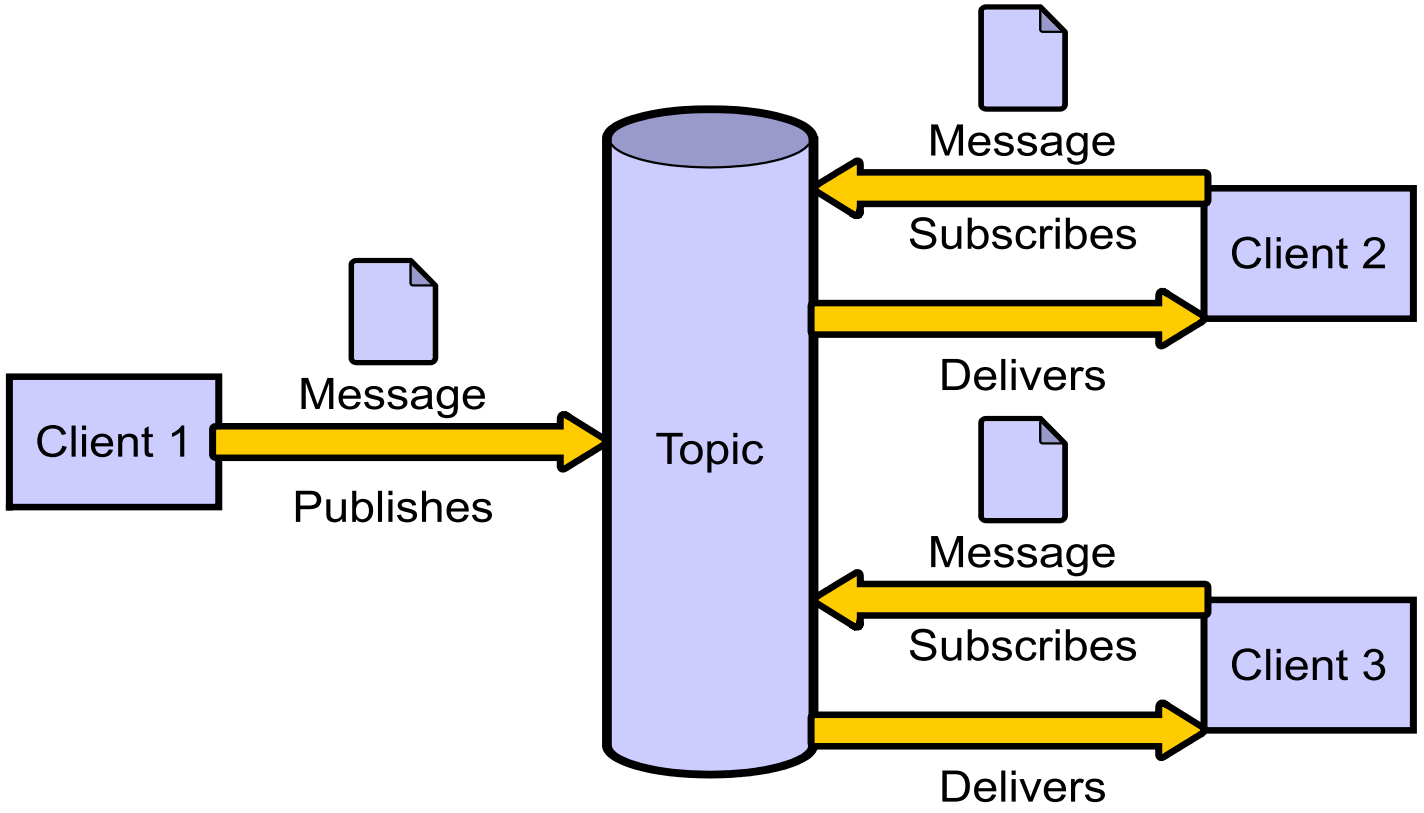


Message Type	Contents of the Message Body
`TextMessage`	A `java.lang.String` object
`MapMessage`	A set of name–value pairs, for example, a hash table
`BytesMessage`	A stream of uninterpreted bytes or binary data
`StreamMessage`	A stream of Java technology primitive values filled and read sequentially
`ObjectMessage`	A serializable Java technology object

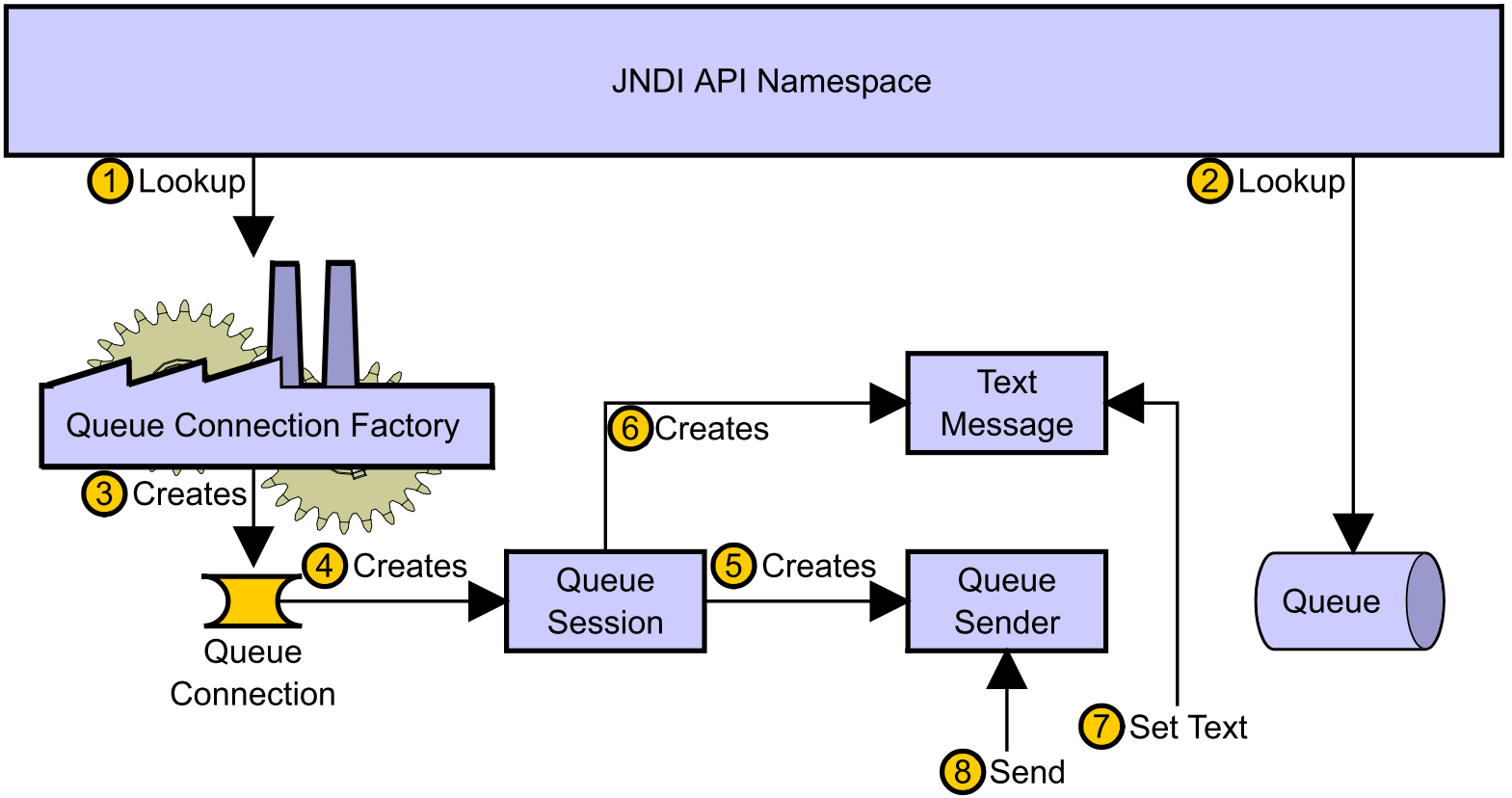
Point-to-Point Messaging Architecture



Publish/Subscribe Messaging Architecture



Creating a Queue Message Producer



Creating a Queue Message Producer

1. Obtain a connection factory using injection or the JNDI API.
2. Obtain the message queue using injection or the JNDI API.
3. Create a `Connection` object using the connection factory.
4. Create a `Session` object using the `Connection` object.
5. Create a `QueueSender` object using the `Session` object.
6. Create one or more `Message` objects using the `Session` object.

Creating a Queue Message Producer

7. Populate message with text using the `setText` method of the `TextMessage` object.
8. Send one or more `Message` objects using the `QueueSender` object.

Example Code for Queue Message Producer

```
1   import javax.jms.*;
2   import javax.naming.*;
3
4   public class SimpleQSender {
5     public static void main(String[] args) {
6       String                  queueName = null;
7       Context                 jndiContext = null;
8       QueueConnectionFactory  queueConnectionFactory = null;
9       QueueConnection         queueConnection = null;
10      QueueSession            queueSession = null;
11      Queue                   queue = null;
12      QueueSender             queueSender = null;
13      TextMessage             message = null;
14      int                     NUM_MSGS = 5;
15
```

Example Code for Queue Message Producer

```
16      // Read queue name from command line and display it.
17       if (args.length != 1) {
18         System.out.println("Usage: java SimpleQSender " +
19           "<queue-name> ");
20         System.exit(1);
21       }
22       queueName = new String(args[0]);
23       System.out.println("Queue name is " + queueName);
24
25       // Create a JNDI API InitialContext object if none exists yet
26       try {
27         jndiContext = new InitialContext();
28       } catch (NamingException e) {
29         System.out.println("Could not create JNDI API " +
30           "context: " + e.getMessage());
31         System.exit(1);
32       }
33
```

Example Code for Queue Message Producer

```
34      // Look up connection factory and queue.  If either does
35      try {
36        queueConnectionFactory = (QueueConnectionFactory)
37          jndiContext.lookup("QueueConnectionFactory");
38        queue = (Queue) jndiContext.lookup(queueName);
39      } catch (NamingException e) {
40        System.out.println("JNDI API lookup failed: " +
41          e.getMessage());
42        System.exit(1);
43      }
44
45      /*
46       * Create connection.
47       * Create session from connection; false means session is
48       * not transacted.
49       * Create sender and text message.
50       * Send messages, varying text slightly.
51       * Send end-of-messages message.
```

Example Code for Queue Message Producer

```
52      * Finally, close connection.
53      */
54      try {
55        queueConnection =
56          queueConnectionFactory.createQueueConnection();
57        queueSession =
58          queueConnection.createQueueSession(false,
59          Session.AUTO_ACKNOWLEDGE);
60        queueSender = queueSession.createSender(queue);
61        message = queueSession.createTextMessage();
62        for (int i = 0; i < NUM_MSGS; i++) {
63          message.setText("This is message " + (i + 1));
64          System.out.println("Sending message: " +
65            message.getText());
66          queueSender.send(message);
67        }
68
```

Example Code for Queue Message Producer

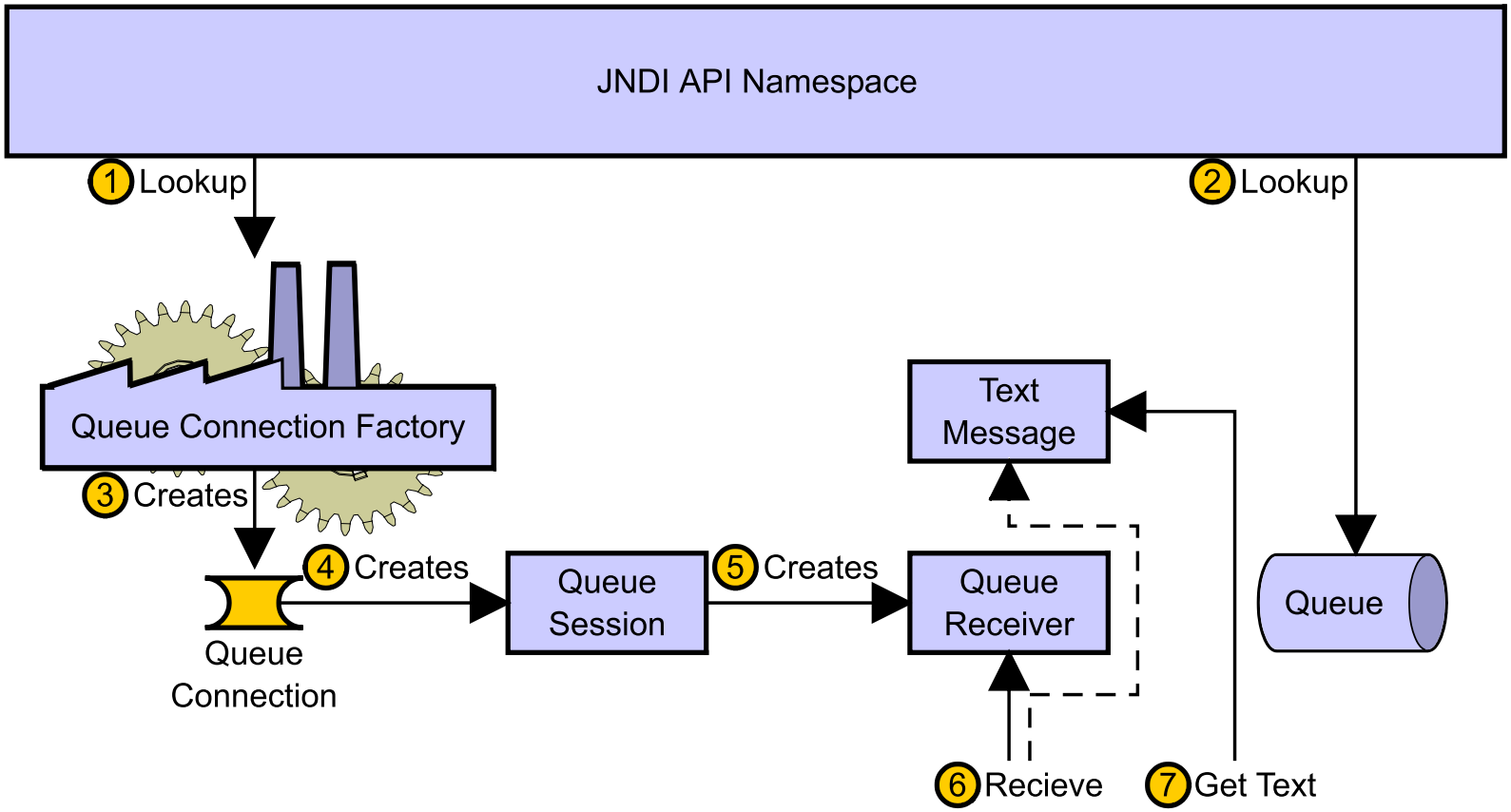
```
69          // Send a non-text control message indicating end of
70          // messages.
71          queueSender.send(queueSession.createMessage());
72        } catch (JMSException e) {
73          System.out.println("Exception occurred: " + e.getMessage());
74        } finally {
75          if (queueConnection != null) {
76            try {
77              Thread.sleep(60000); // delay before closing
78              queueConnection.close();
79            } catch (JMSException e) {}
80          }
81        }
82      }
83    }
84
```

An Alternative to JNDI Lookup: Using Injection

```
@Resource(name="jms/QueueConnectionFactory")
  javax.jms.QueueConnectionFactory queueConnectionFactory;

@Resource(name="jms/someQueue")
  javax.jms.Queue queue;
```

Synchronous Queue Consumer



Synchronous Queue Consumer

1. Obtain a connection factory using injection or the JNDI API.
2. Obtain the message queue using injection or the JNDI API.
3. Create a `Connection` object using the factory.
4. Create a `Session` object.
5. Create a `MessageReceiver` object.

Synchronous Queue Consumer

6. Call the `receive` method of the `MessageReceiver` object.

   The `receive` method blocks if the queue is empty.

7. Process the message returned by the `receive` method.

Example Code for Synchronous Queue Consumer

```
1   import javax.jms.*;
2   import javax.naming.*;
3
4   public class SimpleQReceiver {
5     public static void main(String[] args) {
6       String                  queueName = null;
7       Context                 jndiContext = null;
8       QueueConnectionFactory  queueConnectionFactory = null;
9       QueueConnection         queueConnection = null;
10      QueueSession            queueSession = null;
11      Queue                   queue = null;
12      QueueReceiver           queueReceiver = null;
13      TextMessage             message = null;
14
15      // Read queue name from command line and display it.
16      if (args.length != 1) {
17        System.out.println("Usage: java " +
18                  "SimpleQueueReceiver <queue-name>");
```

Example Code for Synchronous Queue Consumer

```
19      System.exit(1);
20      }
21      queueName = new String(args[0]);
22      System.out.println("Queue name is " + queueName);
23
24      // Create a JNDI API InitialContext object if none exists yet
25      try {
26        jndiContext = new InitialContext();
27      } catch (NamingException e) {
28        System.out.println("Could not create JNDI API " +
29          "context: " + e.getMessage());
30        System.exit(1);
31      }
32
33      // Look up connection factory and queue.  If either does
34      try {
35        queueConnectionFactory = (QueueConnectionFactory)
36          jndiContext.lookup("QueueConnectionFactory");
```

Example Code for Synchronous Queue Consumer

```
37          queue = (Queue) jndiContext.lookup(queueName);
38        } catch (NamingException e) {
39          System.out.println("JNDI API lookup failed: " +
40            e.getMessage());
41          System.exit(1);
42        }
43
44        /*
45         * Create connection.
46         * Create session from connection; false means session is
47         * not transacted.
48         * Create receiver, then start message delivery.
49         * Receive all text messages from queue until
50         * a non-text message is received indicating end of
51         * message stream.
52         * Close connection.
53         */
54        try {
```

Example Code for Synchronous Queue Consumer

```
55          queueConnection =
56            queueConnectionFactory.createQueueConnection();
57          queueSession =
58            queueConnection.createQueueSession(false,
59          Session.AUTO_ACKNOWLEDGE);
60          queueReceiver = queueSession.createReceiver(queue);
61          queueConnection.start();
62          while (true) {
63            Message m = queueReceiver.receive(1);
64            if (m != null) {
65              if (m instanceof TextMessage) {
66                message = (TextMessage) m;
67                System.out.println("Reading message: " +
68                  message.getText());
69              } else {
70                break;
71              }
72            }
```

Example Code for Synchronous Queue Consumer

```
73        }
74      } catch (JMSException e) {
75        System.out.println("Exception occurred: " + e.getMessage());
76      } finally {
77        if (queueConnection != null) {
78          try {
79            queueConnection.close();
80          } catch (JMSException e) {}
81        }
82      }
83    }
84  }
85
```

Creating an Asynchronous Queue Consumer

1. Look up a connection factory using the JNDI API.
2. Look up the message queue using the JNDI API.
3. Create a `Connection` object using the factory.
4. Create a `Session` object.
5. Create a `QueueReceiver` object.
6. Call the `setMessageListener` method of the `QueueReceiver` object passing it an instance of a `MessageListener` interface.
7. Process the received message in the `onMessage` method of the `MessageListener` instance.

Example Code for Asynchronous Queue Consumer

```
1   import javax.jms.*;
2   import javax.naming.*;
3
4   public class SimpleMessageListener implements MessageListener {
5     public void onMessage(Message message) {
6       TextMessage msg = null;
7       try {
8         if (message instanceof TextMessage) {
9           msg = (TextMessage) message;
10          System.out.println("Reading message: " +
11            msg.getText());
12        } else {
13          System.out.println("Message of wrong type: " +
14            message.getClass().getName());
15        }
16      } catch (Exception e) {
17        System.out.println("Exception in onMessage(): " +
18          e.getMessage());
19      }
```

Example Code for Asynchronous Queue Consumer

```
20    }
21
22    public static void main(String[] args) {
23      String                  queueName = null;
24      Context                 jndiContext = null;
25      QueueConnectionFactory  queueConnectionFactory = null;
26      QueueConnection         queueConnection = null;
27      QueueSession            queueSession = null;
28      Queue                   queue = null;
29      QueueReceiver           queueReceiver = null;
30      MessageListener         listener = null;
31
32      listener = new SimpleMessageListener();
33      // Read queue name from command line and display it.
34      if (args.length != 1) {
35        System.out.println("Usage: java " +
36                   "SimpleQueueReceiver <queue-name>");
37        System.exit(1);
```

Example Code for Asynchronous Queue Consumer

```
38        }
39      queueName = new String(args[0]);
40      System.out.println("Queue name is " + queueName);
41
42      // Create a JNDI API InitialContext object
43      try {
44        jndiContext = new InitialContext();
45      } catch (NamingException e) {
46        System.out.println("Could not create JNDI API " +
47          "context: " + e.getMessage());
48        System.exit(1);
49      }
50
51      // Look up connection factory and queue
52      try {
53        queueConnectionFactory = (QueueConnectionFactory)
54          jndiContext.lookup("QueueConnectionFactory");
55        queue = (Queue) jndiContext.lookup(queueName);
```

Example Code for Asynchronous Queue Consumer

```
56      } catch (NamingException e) {
57        System.out.println("JNDI API lookup failed: " +
58          e.getMessage());
59        System.exit(1);
60      }
61
62      /*
63       * Create connection.
64       * Create session from connection; false means session is
65       * not transacted.
66       * Create receiver, then start message delivery.
67       * Receive all text messages from queue until
68       * a non-text message is received indicating end of
69       * message stream.
70       * Close connection.
71       */
72      try {
73        queueConnection =
```

Example Code for Asynchronous Queue Consumer

```
74      queueConnectionFactory.createQueueConnection();
75      queueSession =
76          queueConnection.createQueueSession(false,
77        Session.AUTO_ACKNOWLEDGE);
78        queueReceiver = queueSession.createReceiver(queue);
79        queueReceiver.setMessageListener(listener);
80        queueConnection.start();
81      } catch (JMSException e) {
82        System.out.println("Exception occurred: " + e.getMessage());
83      } finally {
84        if (queueConnection != null) {
85          try {
86            queueConnection.close();
87          } catch (JMSException e) {}
88        }
89      }
90    }
91  }
```

Evaluating the Capabilities and Limitations of EJB Components as Messaging Clients

Enterprise Beans	Producer	Synchronous Consumer	Asynchronous Consumer
Session bean	Yes	Not recommended	Not possible
Message-driven bean	Yes	Not recommended	Yes

- Only message-driven beans can implement the `javax.jms.MessageListener` interface.
- Entity and session beans participating in messaging are required to include the standard JMS API messaging producer or synchronous consumer code.

Summary

The JMS API provides an abstraction on top of a messaging system. The two messaging models are a point-to-point model using Queues and a publish-subscribe model using Topics. Just about any component can produce messages but because of the strict thread management enforced by an application server, not all components are suited to consume messages.