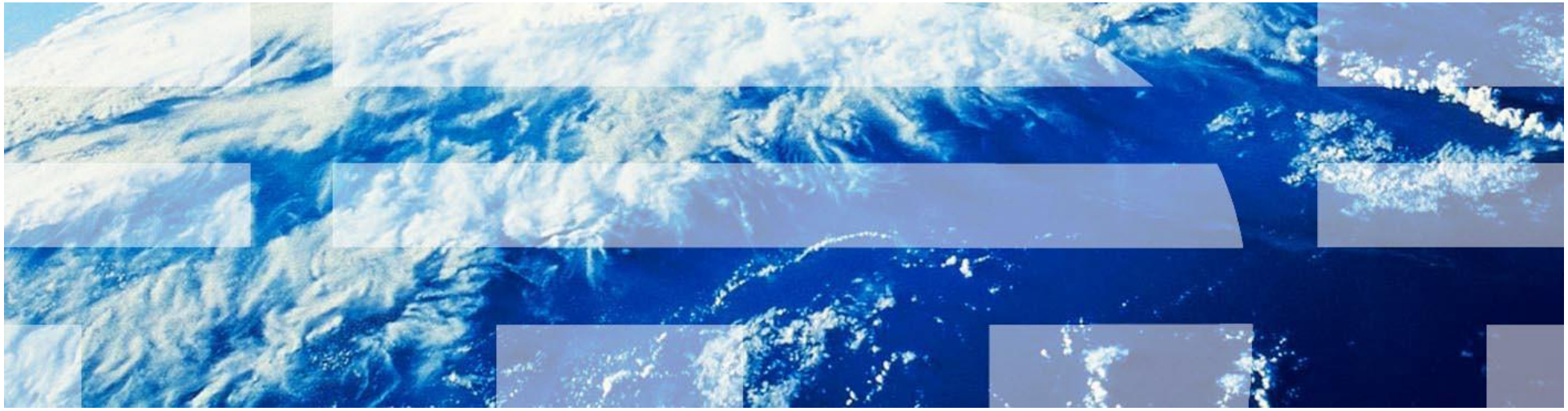
IBM MQ

Overview of the IBM MQ JMS Provider

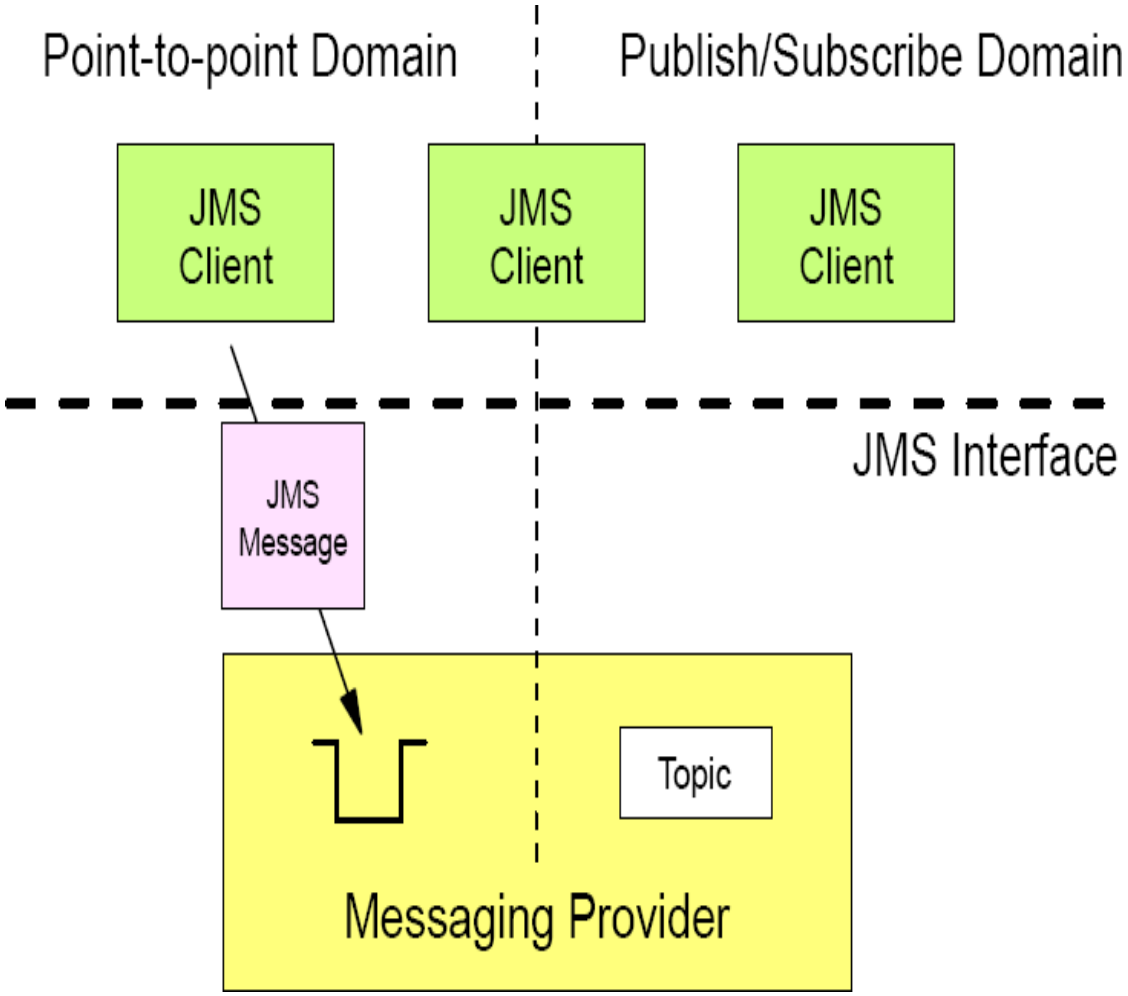


Unit Agenda

- Basic Concepts of the Java™ Message Service (JMS)
  기본 개념
  - What is it? Why use it?
  - What are:
    - Administered objects
      - Connection Factories
      - Destinations
    - Connections
    - Sessions
    - Message Producers and Consumers
- IBM® MQ 의 JMS 구현 방식
- JMS Provider 로 IBM MQ 의 장점
- Lab 2 - IBM MQ JMS Provider

What is the Java Message Service (JMS)?

- JMS 는 고정된 API ,JCP 에 의해 관리
- JMS API 는 인터페이스 집합으로 제공
- Java 프로그래머를 위한 통합된 메시징 접근 방식 제공



Notes - What is the Java Message Service (JMS)?

N

O

T

E

S

Java Community Process

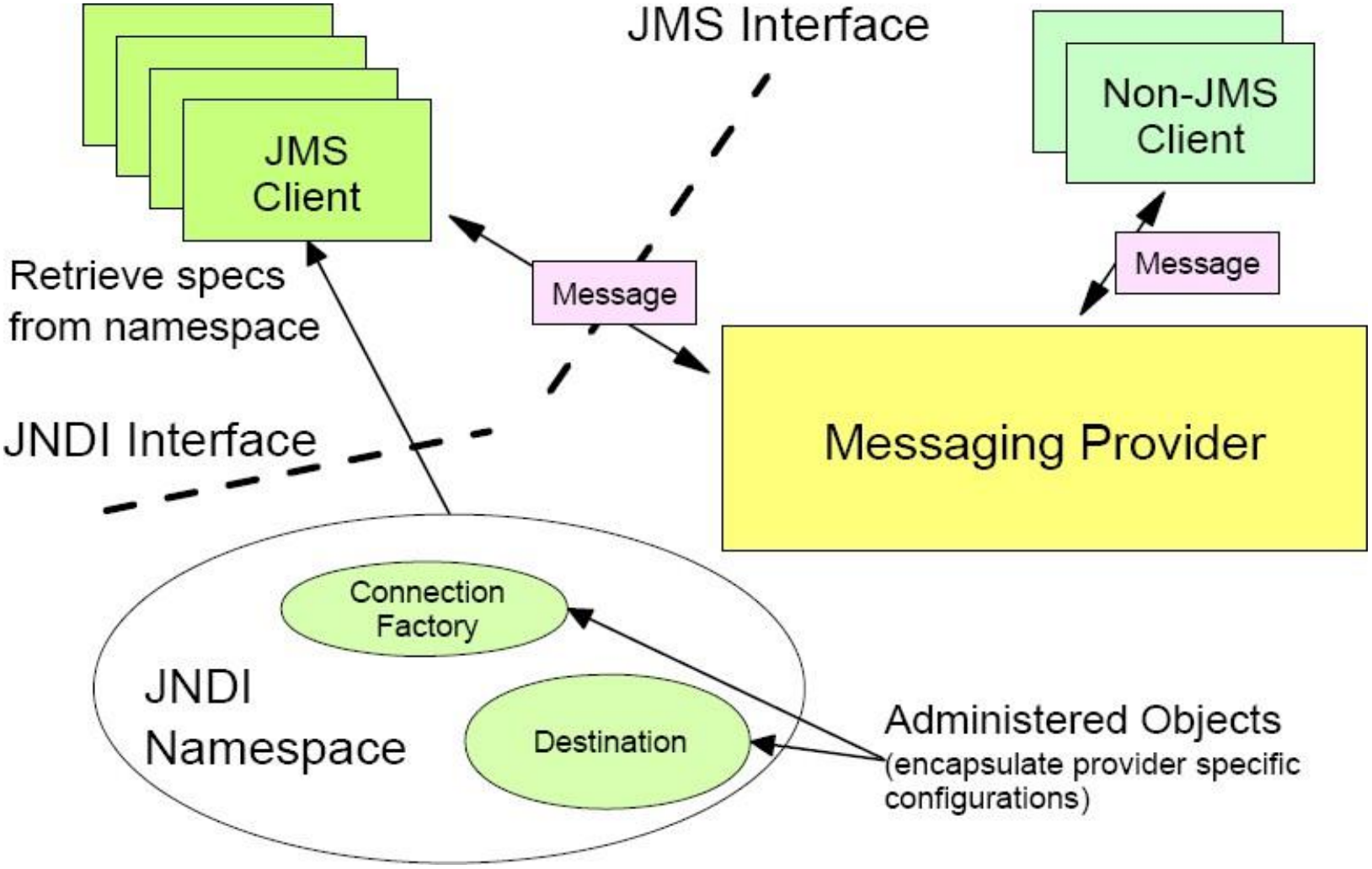
JMS was initially defined by Sun Microsystems in collaboration with Java industry partners, along with other similar specifications such as J2EE, JTA etc. These specifications are typically managed under the Java Community Process which is an industry partnership for defining and agreeing enhancements to such specifications (and new specifications for Java technologies).

Programs written with JMS can run on any messaging system that implements the JMS standard.

The objectives of JMS

- JMS 의 주요 목적
  - 공통 메시징 개념 및 기능 정의
  - 학습 곡선 최소화
  - 애플리케이션 이식성 극대화
  - 제공자 구현 작업 최소화
  - 메시징 도메인에 대한 API 인터페이스 제공
    - Point-to-point
    - Publish/Subscribe

JMS Architecture



Notes - JMS Architecture

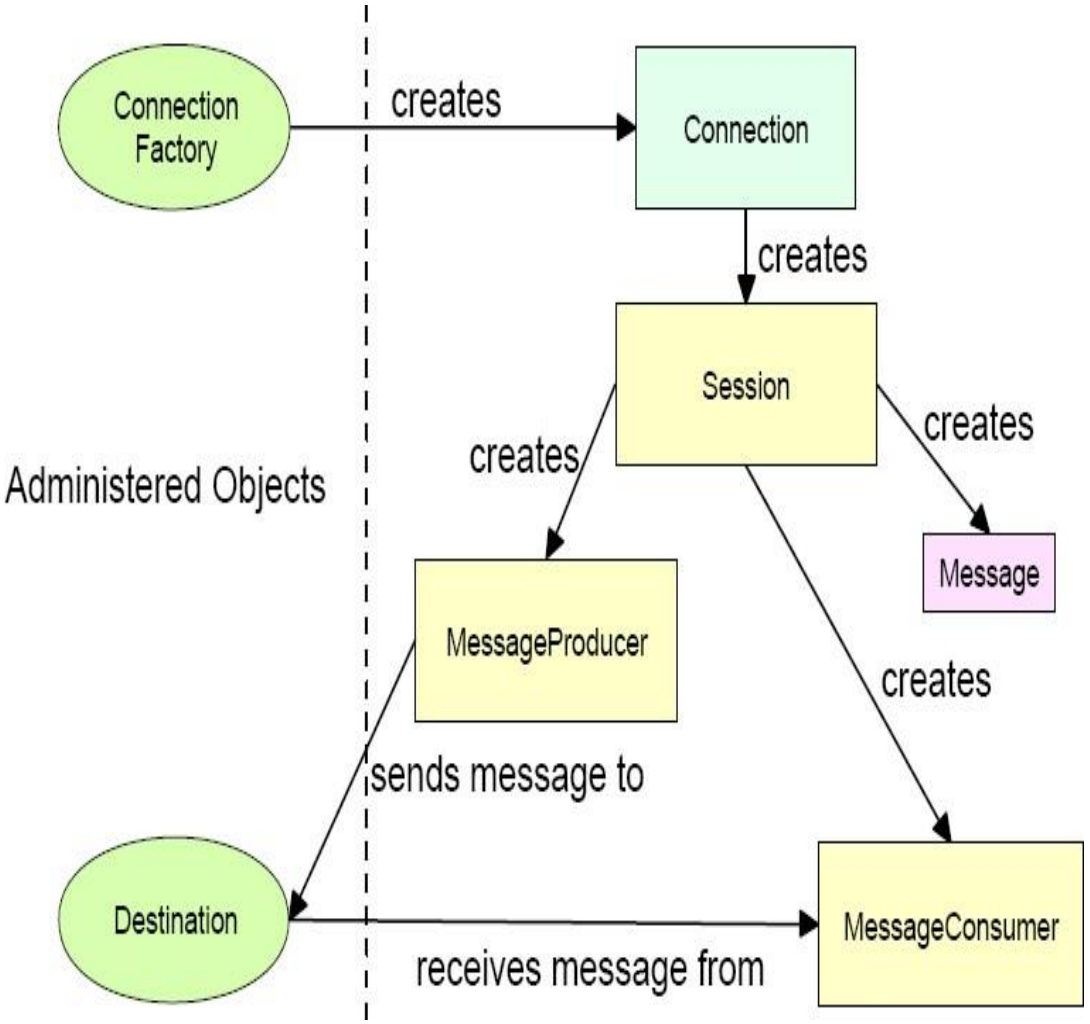
N O T E S

The details of each component in the diagram are presented in the next slide.

The JMS Client represents the application. The JNDI Namespace is for the Administered objects. The Messaging Provider is for example IBM MQ.

JMS Classic API 의 구성요소

- ***Connection Factory***:..관리객체, 메시징 시스템에 연결 생성
- ***Connection***: 프로바이더와의 활성 연결 .
- ***Session***: 메시지를 보내고 받기 위한 단일 스레드 컨텍스트
- ***Message Producer***: 메시지 전송자
- ***Message Consumer***: 메시지 수신자
- ***Destination***: 메시지의 수/발신 의 목적지를 캡슈롸 하는 관리 객체



Notes - Building Blocks of JMS Classic API

N

O

T

E

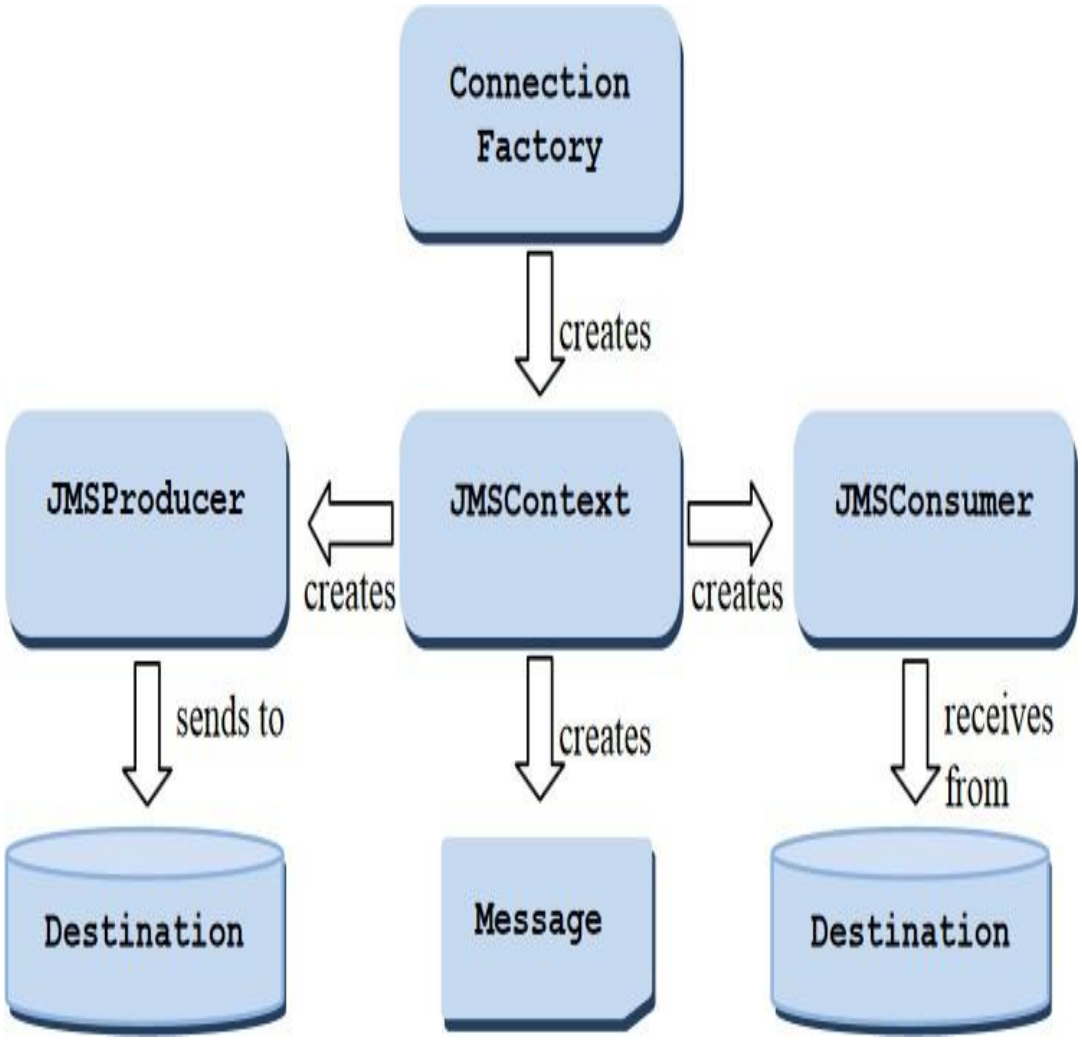
S

Explaining the diagram in IBM MQ terms:
Connection provides a place to hold the parameters that control how to connect to MQ, eg QMgr name, hostname for remote connections, etc

Destination defines a queue or topic
Session defines a HCONN and therefore a transactional scope.
Producer/Consumer contains a HOBJ for putting/getting

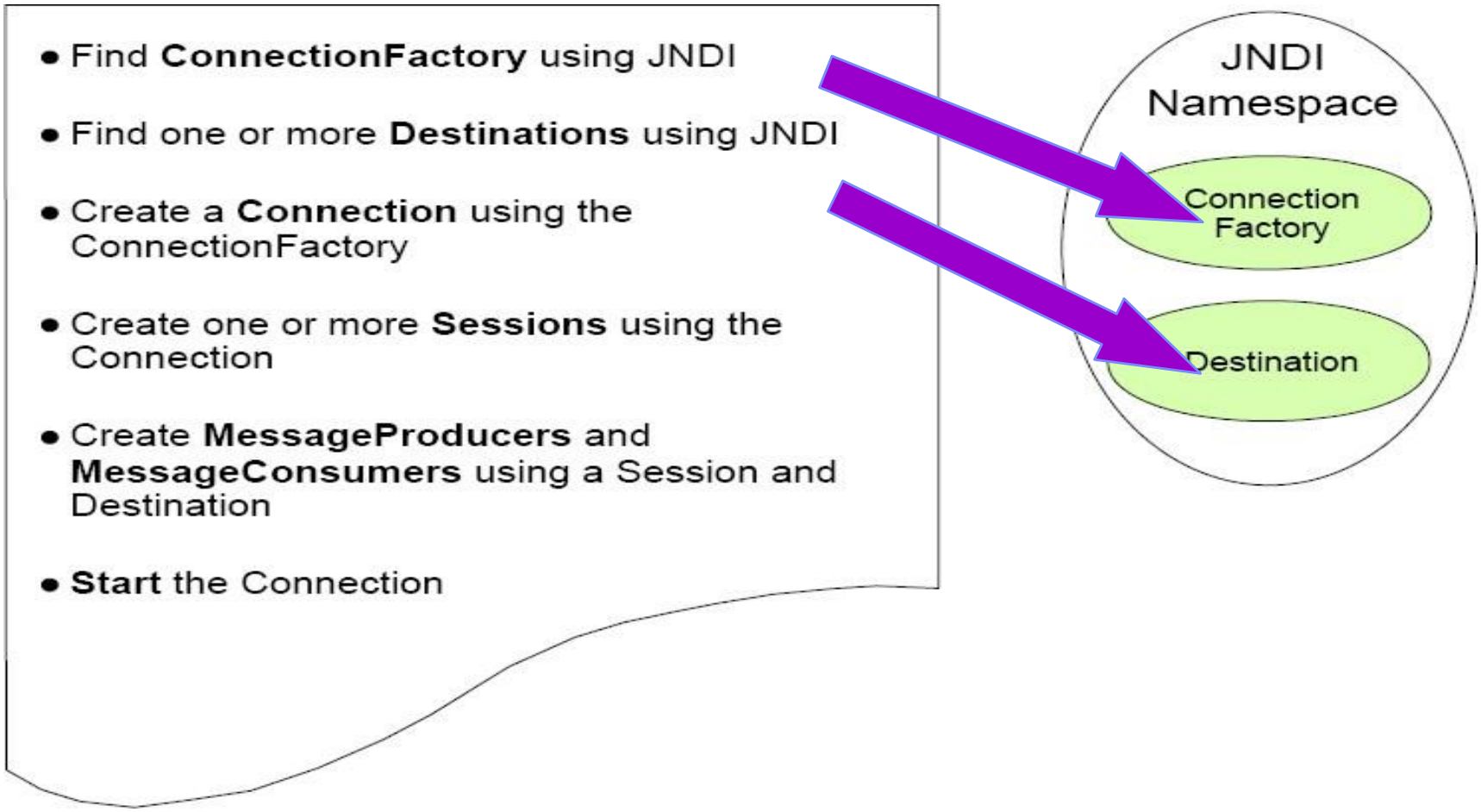
JMS Simplified API 구성요소

- ***Connection Factory***: 관리 객체로 메시징 시스템에 연결 생성
- ***JMSContext*** JMS 프로바이더와의 활성 연결 및 메시지를 송수신하기 위한 단일 스레드 컨텍스트
- ***JMS Producer*** – JMSContext 에 의해 생성된 객체로 큐나 토픽으로 메시지를 전송
- ***JMS Consumer*** – JMSContext 에 의해 생성된 객체로 큐나 토픽에 보내진 메시지를 수신



Developing a JMS program with Classic API

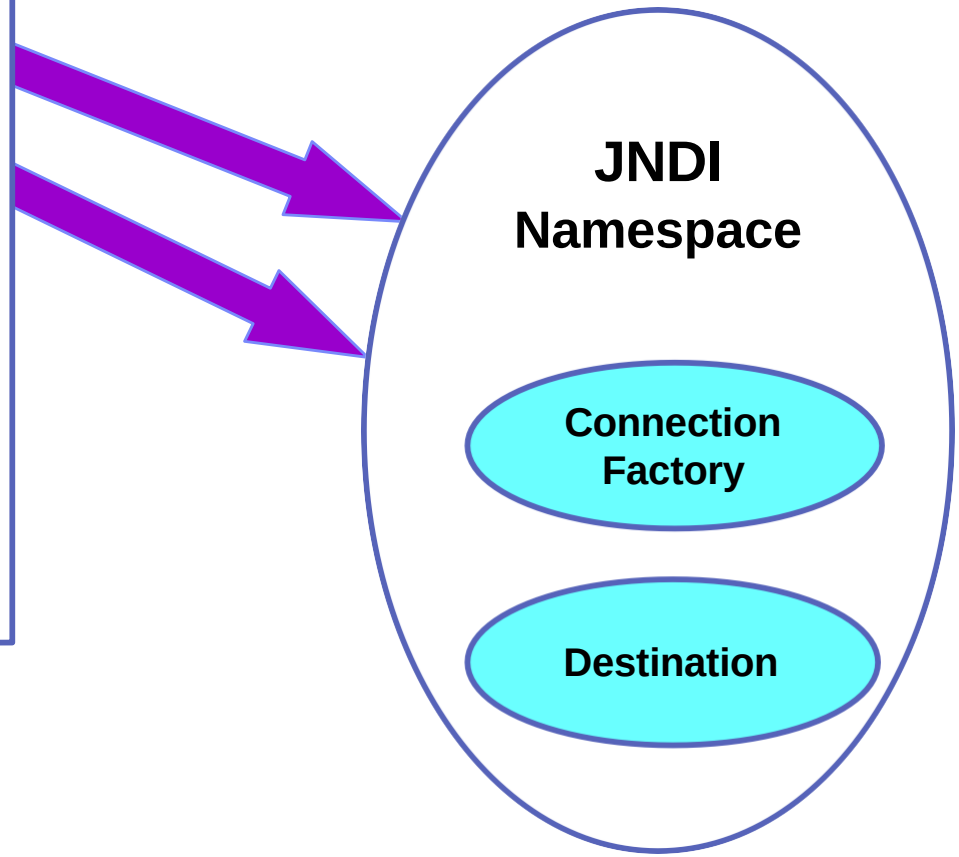
- 일반적인 메시지 처리 흐름 :



Developing a JMS program with Simplified API

- 일반적인 메시지 처리 흐름

- Find **ConnectionFactory** using JNDI
- Find one or more **Destinations** using JNDI
- Use **ConnectionFactory** to create a **JMSContext** object
- Use the **JMSContext** object to create the **JMSProducer** and **JMSConsumer** objects needed
- Delivery of messages begins automatically



JMS Messages

Header	• Used to identify and route messages • All messages support the same set of header fields
Properties	• Used to add optional header fields to a message • Categories: – JMS optional header fields – Application-specific properties – Provider-specific properties
Body	• Actual message data to be delivered • Categories: – Text – Stream – Map – Bytes – Object

Notes - JMS Messages

N O T E S

Header All messages support the same set of header fields. Header fields contain values that are used by both clients and providers to identify and route messages.
Properties Each message contains a built-in facility to support application-defined property values. Properties provide an efficient mechanism to filter application-defined messages.
Body JMS defines several types of message body that cover the majority of messaging styles currently in use. JMS defines five types of message body:
Text A message containing a java.lang.String.
Stream A stream of Java primitive values. It is filled and read sequentially.
Map A set of name-value pairs, where names are strings and values are Java primitive types. The entries can be accessed sequentially or randomly by name. The order of the entries is undefined.
Bytes A stream of uninterpreted bytes. This message type is for literally encoding a body to match an existing message format.
Object a message that contains a serializable Java object

IBM MQ as a JMS Provider

IBM MQ as a JMS Provider

- IBM provides an implementation of JMS over IBM MQ
  - IBM은 JMS를 IBM MQ 위에서 구현한 **IBM MQ classes for Java Message Service(IBM MQ JMS)**를 제공
  - 이를 통해 IBM MQ를 사용하는 애플리케이션이 JMS API를 활용해 메시지를 주고받을 수 있습니다.
- The IBM MQ JMS implementation has evolved over time
  - IBM MQ JMS는 시간이 지나면서 크게 발전
  - JMS 계층의 개선: JMS API 자체가 성능 및 사용성을 개선하며 발전
  - IBM MQ의 새로운 기능: IBM MQ 자체도 새로운 기능과 성능 최적화를 통해 더욱 발전
  - 이러한 발전은 주로 성능 향상과 사용성 개선으로 나타납니다.

Notes - IBM MQ as a JMS Provider

N O T E S

Wire Format

MQ JMS

The MQ JMS implementation, aka MQ Classes for JMS, is that defined within com.ibm.mq.jms.*, not to be confused with the MQ Classes for Java in com.ibm.mq.*, which is a Java OO form of the MQI.

Performance

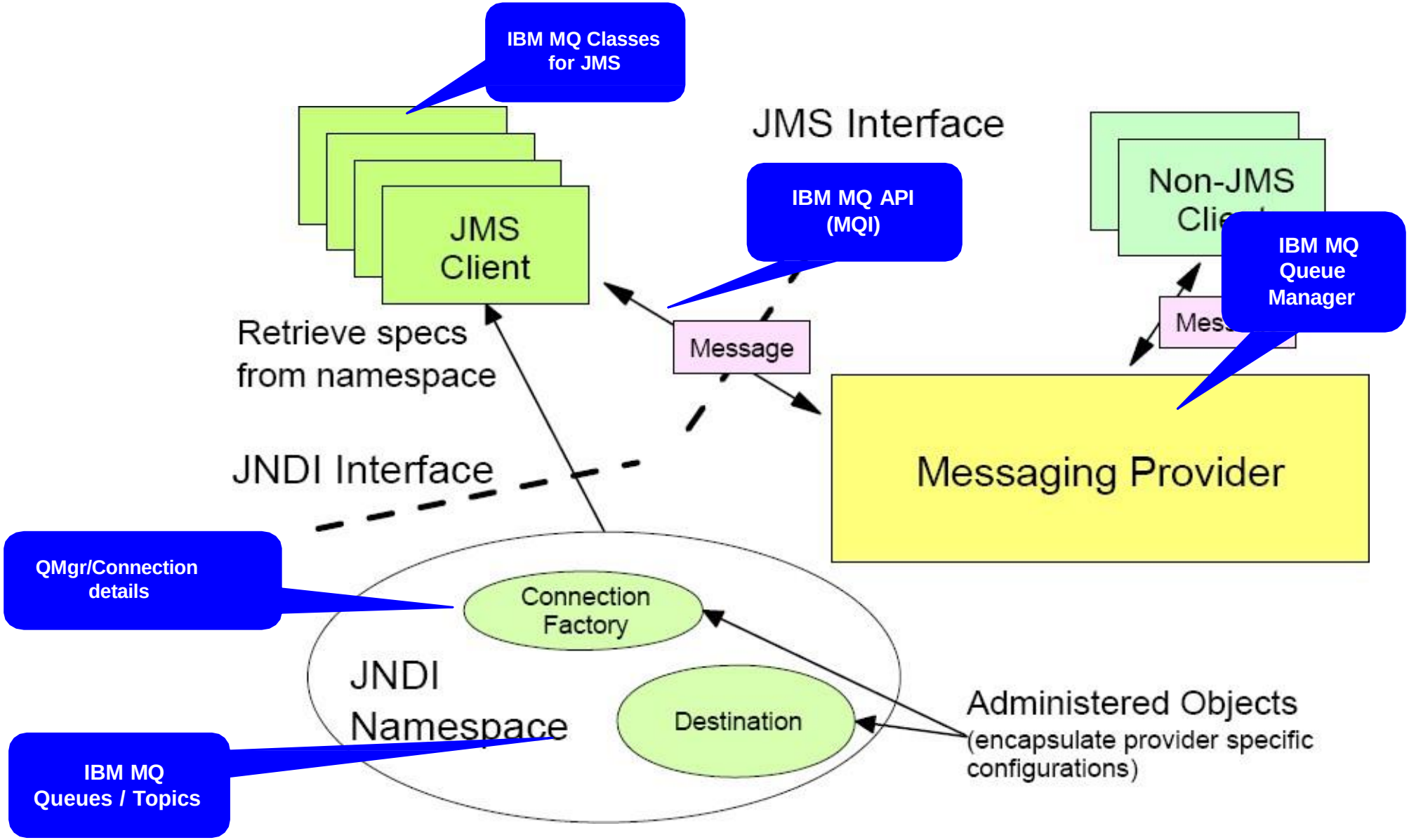
By ‘performance improvements’, we mean any enhancement that improves the runtime behaviour of the MQ JMS implementation in terms of the efficient use of resources. Typically these would be seen as increased message throughput, enhanced scalability to more client connections, reduced network socket or bandwidth usage, reduced CPU or memory usage etc.

Usability

By ‘usability improvements’, we mean any enhancement to the ease of use of MQ by JMS developers, administrators etc. These may be in installation, configuration, documentation, administration and monitoring tools, development tools, problem determination assistance, samples, demonstration applications etc.

In JMS implementations, the wire format of messages is typically proprietary. JMS implementations should use a common API but should not be expected to interoperate with each other, ie the JMS spec is for portability and not interoperability.

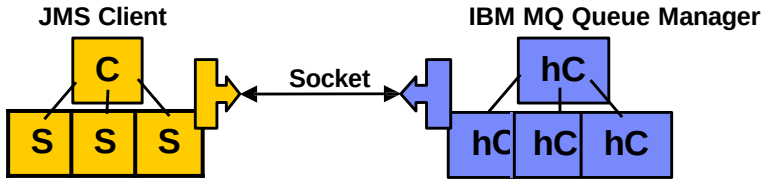
JMS Architecture mapped to MQ objects



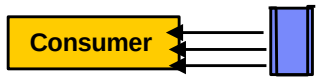
JMS Constructs Mapped to IBM MQ Features

The IBM MQ Classes for JMS exploit a number of performance and interoperability features available to applications using the Native MQ API, including:

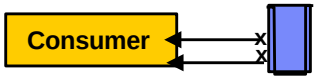
– 다중 연결 : Multiplexed Connections



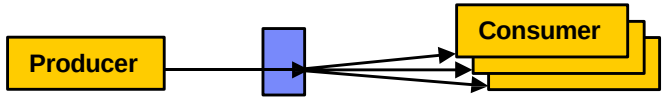
– 비동기 소비자 : Asynchronous Consumers



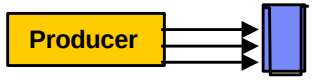
– Selectors



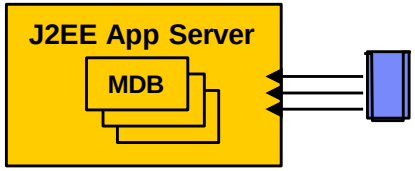
– Publish/subscribe, as well as point-to-point messaging



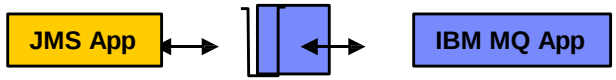
– Non-blocking Producers



– Read-Ahead



– Interoperability between JMS and other IBM MQ applications



Notes - JMS Constructs Mapped to IBM MQ Features

N

O

T

E

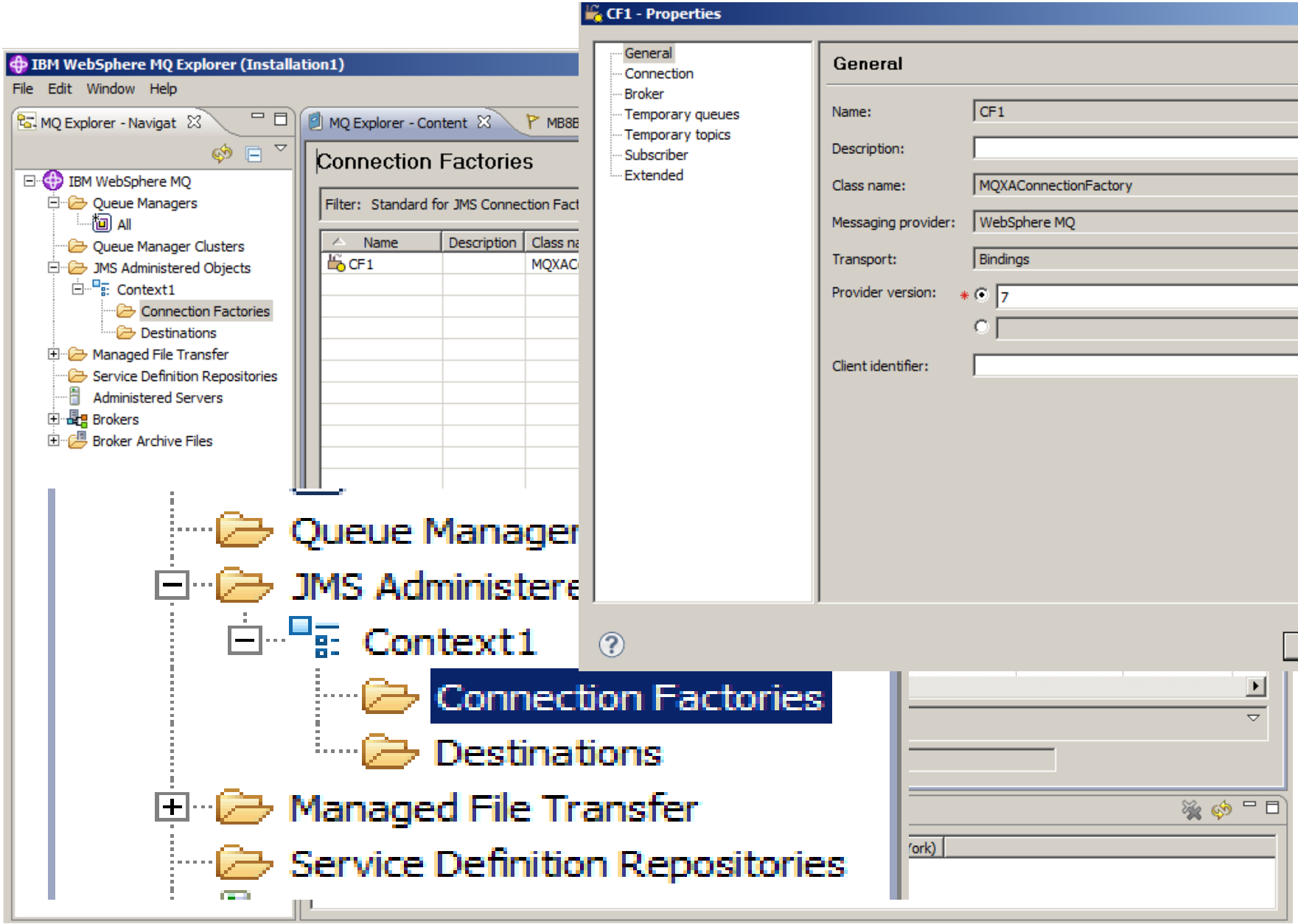
S

High level - details of each point explained in the following slides

A set of features to be implemented in IBM MQ v7.5.5 will enhance the performance and usability of the product as a JMS provider. This set of features is being implemented under a series of DCRs to make modifications to the main product implementation and API, in addition to a series of DCRs to provide a new JMS (and XMS for C/C++) implementation making use of these underlying product enhancements.

This will provide equivalent functionality to the previous implementation used in earlier releases of the product, so as to both meet the JMS Specification and ensure that applications migrated from earlier releases to this new implementation continue to behave in the same way. The purpose of the new implementation is to enhance usability of IBM MQ when using the JMS API and/or enhance runtime performance, not to provide additional functionality.

JMS Administration using the MQ Explorer



IBM MQ JMS Provider 의 개발자 혜택

- 풍부한 샘플 제공
  - Specific, documented samples for point-to-point and publish/subscribe producers and consumers
- MQ Explorer 통합 제공
  - Administrative view of IBM MQ as a JMS provider
- 간편한 “즉시사용“ 환경
  - Default configuration optimized for use as a JMS provider
- 문제해결 및 서비스 지원 도구
  - e.g. JMS trace facilities provide detail diagnostics

Notes - Benefits of the IBM MQ JMS Provider for Developers

N O T E S

Integration with MQ Explorer – can automatically create a queue for example when defining a destination in the Administered Objects section of Explorer and vice/versa when defining a queue you can created a Destination. In both cases a wizard pops up to complete the details.

Enhanced JMS trace facilities allows better support – from IBM as well.

IBM MQ JMS Provider 의 System Administrators 혜택

- MQ Explorer 통합
  - Administrative view of IBM MQ as a JMS provider
- 통합된 publish/subscribe 엔진
  - Started, configured and monitored as part of the queue manager
  - No need for complex control, publication and subscription queues
- Queue 와 Topic 관리 및 보안
  - Access to both can be configured and managed
- 간단하고, 직관적인 구성 및 튜닝
  - Default configuration optimized for use as a JMS Provider
  - Simple configuration management
    - “One-step” MQ and JMS/JNDI** configuration for queues and topics
- 강력한 문제 해결 및 서비스 지원 도구

** Java Naming and Directory Interface

IBM MQ JMS Provider 의 Operations Managers 혜택

- 높은 처리량 제공

  - Latest version provides improvements in non-blocking producers, client read-ahead, asynchronous consumers, selector support, message driven beans (MDBs), topic subscribers
- 네트워크 소켓 사용 최소화
  - Multiple Connections and/or Sessions can share the same socket
- 네트워크 대역폭 절감
  - Most processing performed server-side, reducing network I/O
  - Optimized control message flows, e.g. for topic subscription
- 최소한의 메모리 사용
  - Lightweight client-side implementation minimizes memory usage
- 최소한의 CPU 사용
  - No need for client-side queue polling, most processing performed server-side
- 간단한 클라이언트 배포 및 구성
  - Default configuration optimized for JMS
- 간소화된 Pub/Sub 관리
  - Common administrative model for JMS and native MQ Publish/Subscribe