

**Chapter  
1**

**INTRODUCTION**

*Get on the  
Fast Track!*



**SYS-ED/  
Computer  
Education  
Techniques, Inc.**

## Objectives

You will learn:

- WebSphere MQ: objects.
- MQSeries concepts and terminology.
- Advantages and problems associated with asynchronous processing.
- Triggers and events.
- One and two way communications.

## **1 MQSeries: What is it**

Middleware is an intermediate software component that bridges dissimilar computing environments.

MQSeries is a middleware product that implements a messaging and queuing framework.

### **1.1 Messaging and Queuing**

Messaging	Programs which communicate by sending data in messages rather than by calling each other directly.
Queuing	Messages are put on queues in storage, eliminating the need for programs to be logically connected.

A messaging and queuing framework is inherently asynchronous.

### **1.2 MQSeries History**

1992	Systems Strategies (SSI) develops ezBridge, a messaging and queuing product for VMS, Tandem, and Unix.
1992	IBM announces Networking Blueprint defining three standard APIs for program to program communication: CPI-C, RPC, MQI.
1992-3	State Street Bank (Boston) evaluates IBM messaging product (code name "Victory") for IBM CICS/ESA and SSI's ezBridge on VMS and Tandem.
1993	IBM buys intellectual property rights for ezBridge from SSI. Initially, IBM's version of MQSeries ran only on mainframe (CICS/ESA, IMS/ESA, and eventually VSE). SSI version (called MQSeries Version 1) initially released on: VMS, Tandem, AS/400, and Unix (SCO, UnixWare).
1994/1995	IBM releases the first three "distributed" platforms: AIX, OS/2, and AS/400.
2004	MQSeries runs on over 35 platforms.

## 2 Asynchronous versus Synchronous Communications

Synchronous:	An application sends a request, then blocks until the request is processed. This requires service availability at exactly the same time as the client needs service.
Asynchronous	An application sends a request and checks at some future time when it is complete. Service does not need to be available when client sends request.

### 2.1 Synchronous

Advantage	Disadvantage
Easy to program.	Better real-time response.
Outcome is known immediately.	Requestor blocks, held resources are "tied up".
Error recovery easier.	Usually requires connection-oriented protocol.
Better real-time response.	

### 2.2 Asynchronous

Advantage	Disadvantage
Requests need not be targeted to specific server.	Response times are unpredictable.
Service need not be available when request is made.	Error handling is usually more complex.
No blocking, so resources could be freed.	Usually requires connection-oriented protocol.
Could use connectionless protocol.	Harder to design applications.

### **3 Messaging versus Procedure Calls**

Messages are an abstract concept; in stark contrast to the procedural mindset of a programmer.

The advantages in using abstractions are:

- Freedom to concentrate on the design of the application.
- There no longer is a need to be concerned with the details of the environment.
- The application becomes portable and extensible.

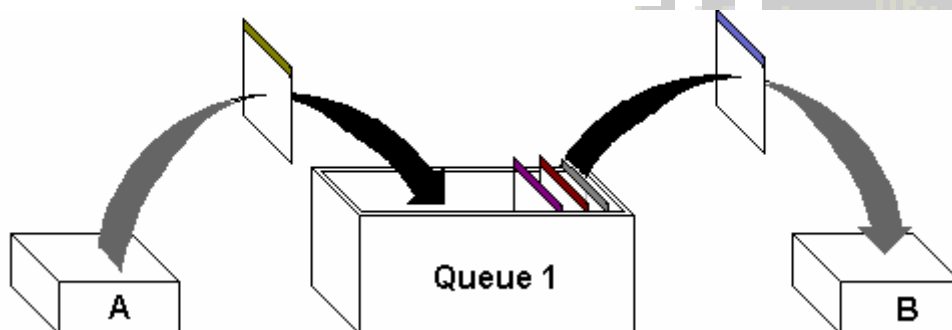
---

## 4 Messaging and Queuing

Programs communicate by putting messages on queues.

Program A puts a message on Queue1, which is read by program B.

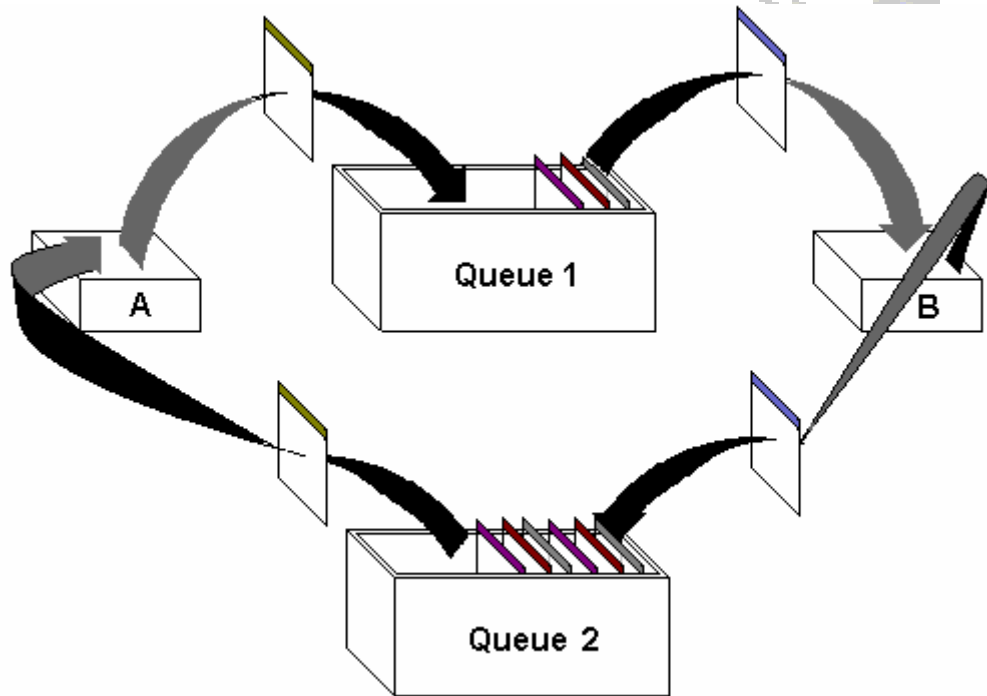
Queue A and B need not be on the same machine.



### 4.1 Two Way Communications

Communication can be one-way or two-way.

A sends to B on Queue1; B responds to A on Queue2



## 4.2 Messaging and Queuing Characteristics

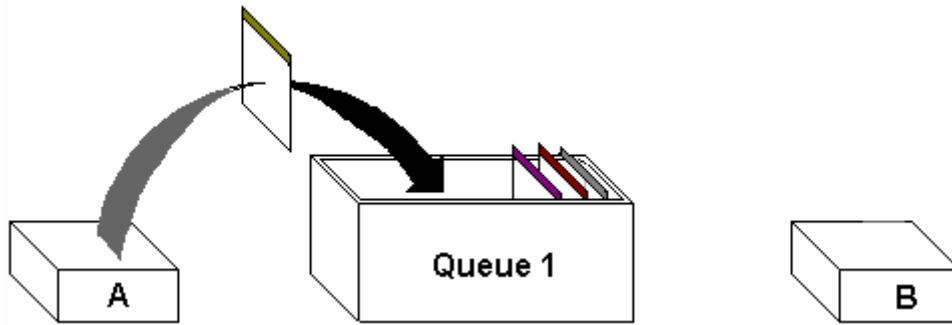
There are three characteristics to messaging and queuing which differentiate it from other communication styles:

- Communicating programs can run at different times.
- There are no constraints on application structure.
- Programs are insulated from environmental differences.

### 4.3 Applications Can Run at Different Times

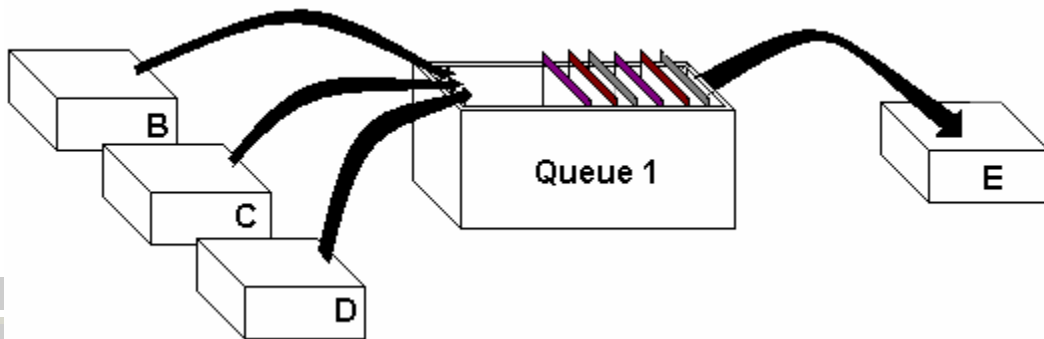
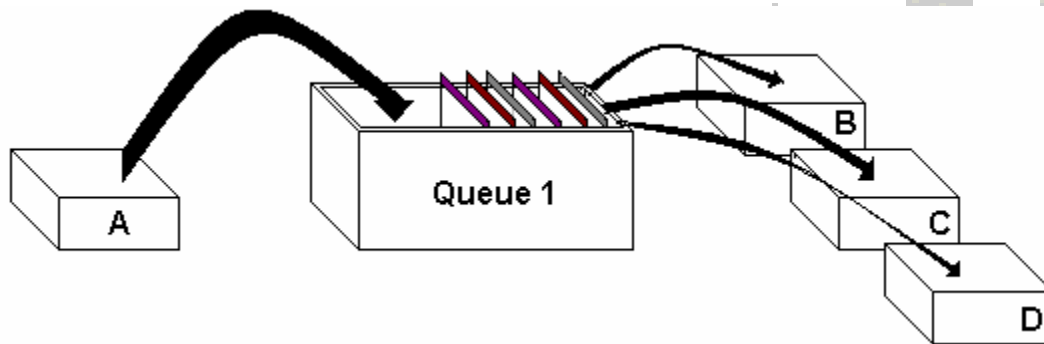
Either program can be unavailable

Message queue exists independently from programs that use them.



### 4.4 No Constraints on Application Structure

There can be a one to many relationship between applications or a many to one relationship between applications.



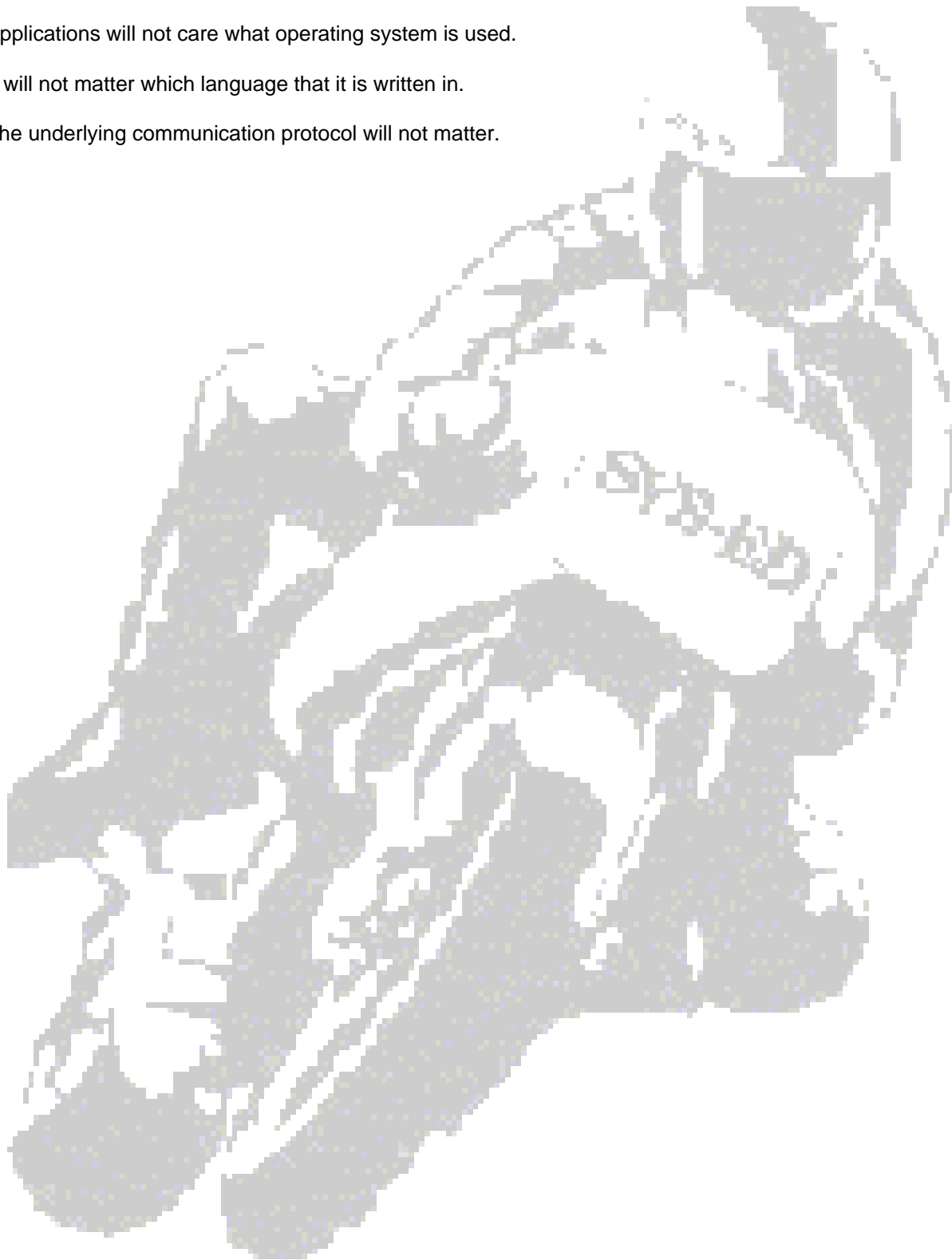
---

## **5 Applications Shielded from Environmental Differences**

Applications will not care what operating system is used.

It will not matter which language that it is written in.

The underlying communication protocol will not matter.



---

## 6 Queue Manager

A Queue Manager controls access to queues:

- administration - create, delete, etc.
- usage - Put, Get
- serves as transaction/syncpoint coordinator for all queue operations.
- access is through the Message Queue Interface – MQI

Queue Managers have names/identities that are UNIQUE in a network such as with host names.

A queue instance is fully qualified by its queue manager and queue name.

Local Queue	An actual queue for which storage is allocated.
Remote Queue	A definition of a queue on a different queue manager which acts somewhat like a pointer.
Alias Queue	Another name for a local or remote queue. Typically used to switch queue destinations without modifying program code.
Model Queue	A template whose properties are copied when creating a new dynamic local queue - " create queue xxx "like" queue yyy.

## **7 Events and Triggering**

Local queues can generate events under certain conditions. These “event” messages can be used to “trigger” the execution of a program.

These events are called trigger messages. The queue on which they are put is called an Initiation Queue.

Process defines an application to an MQSeries queue manager. A process definition object is used for defining applications to be started by a trigger monitor.

A trigger monitor is a program that listens on an initiation queue and executes commands named in Process definitions.

Triggering is useful when there is a requirement not to deploy long-running programs.

## **8 Message Channels**

Message channels provide a communication path between two queue managers on the same, or different, platforms.

A message channel can transmit messages in one direction only. If two-way communication is required between two queue managers, two message channels are required.

There are six types of message channels:

- Sender - initiates connection to Receiver
- Server - Accepts request to start from requester, then becomes Sender
- Receiver - Passive; waits for initiation sequence from Sender
- Requester - Active at start, then becomes Receiver
- Cluster-sender; used amongst Cluster Queue Managers
- Cluster-receiver; used amongst Cluster Queue Managers

## **9 Assured Delivery**

When queues are persistent, and message is persistent, then MCAs will eventually deliver the message to the target queue.

MCAs have recoverable state and a connection-oriented protocol.

Message is not removed from xmit queue until partner MCA confirms placement on the target queue.

---

**10 Message Types**

---

Request	Used by one program to ask another program for something (usually data). A request message needs a reply.
Reply	Used in response to a request message.
One-way message	Does not need a reply, though it can carry data.
Report message	Used when something unexpected occurs, or to give extra information like: <ul style="list-style-type: none"><li>• message delivered to target queue.</li><li>• message taken by application.</li><li>• message not deliverable.</li><li>• message exceeded time-to-live.</li></ul> Messages are added and removed from queues in Units of Work The smallest Unit of Work is one message. Units of work are atomic.
Unit of recovery	A piece of work that changes data from one point of consistency to another.
Syncpoint	A point of consistency which is also called a commit point. It is a moment at which all the recoverable data that an application program accesses is consistent.

---

## 11 Logging and Recovery

All operations that affect the “state” of the Queue Manager and its objects are logged to a log file.

State is:

- Object definitions -queue manager, queues, processes, channels, etc)
- Queue content -messages)

Message channel states are logged separately by each channel

There are two forms of logging:

Circular	Log records are written sequentially across several files, then “wrap” back to the first file.
Linear	Log records are written sequentially across files. New files are allocated as current files fill. There is no automatic reuse of file space.

---

**12 MQSeries Programming Interface**

MQCONN	Connect to queue manager.
MQDISC	Disconnect from queue manager.
MQOPEN	Open queue
MQCLOSE	Close queue
MQPUT	Put message
MQGET	Get message
MQPUT1	Put one message: implied open/put/close
MQBEGIN	Begin unit of work
MQCMIT	Commit
MQBACK	Back out
MQINQ	Inquire about queue attributes
MQSET	Set queue attributes