Appendix B

WORKSHOP

SYS-ED/
Computer
Education
Techniques, Inc.

1 Getting Started

Objectives

- Identify key components of IBM MQ.
- Understand the relationship between the queues and the software.

Questions

1. The message ______________identifies the message and contains additional control information, such as the type of message and priority assigned by the sending application.

2. Queues are managed by a ______________ __________________.

3. The manipulation or administration of objects includes starting and stopping ____________________ _________________________.

4. There are no direct ________________between programs.

5. In the event that recovery is required, all _____________messages will be restored.

6. Each object is identified by an object ____________________.

7. The channel initiator provides and manages resources that enable WebSphere MQ distributed queuing. IBM MQ uses _________________- __________________ _________________ to send messages from one queue manager to another.

8. The ___________ utility program is provided with IBM MQ to help in performing commands and processing object definitions.

9. A simple message for which no reply is expected is a _________________________.

10. A message that describes an event such as the occurrence of an error is a _______________________ .

2 Messages

2.1 Local Queue

Objectives

- Create a local queue based on an existing queue.
- Use the defaults and the attributes from the model queue.
  - The instructor will provide the names of the queue and the queue managers.
- Display the queue with all of its attributes.

Exercise

1. Use this naming convention for the queue name:

   `Firstname.QUEUE`

2. There are a number of methods that can be used to create a queue.

3. In the Microsoft Windows and UNIX environments, use the runmqsc commands.

   This skeleton JCL will be used for running the utility in the z/OS environment.

```
//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1 ')
//STEPLIB DD DSN=thlqual.SCSQANLE,DISP=SHR
//DD DSN=thlqual.SCSQAUTH,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
COMMAND DDNAME(COMMAND)
/*
//COMMAND DD *
DEFINE QL (Firstname.QUEUE) LIKE(instructor_provided)
DISPLAY QL (Firstname.QUEUE) ALL
/*
```

3 WebSphere MQ Objects

Objectives

- Use the CSQUTIL for z/OS or runmqsc for the Microsoft Windows / UNIX platforms to display system information.
- Display information about queues, channels, processes, and logs.

Exercise

1. Use MSQC commands, to display this information:

   1.1. archive information

   1.2. channel definition

   1.3. channel status

   1.4. command server

   1.5. channel initiator

   1.6. active log information

   1.7. Namelist

   1.8. Processes

   1.9. queue manager information

   1.10. status of a queue

   1.11. queue information

   1.12. active and in-doubt threads

4 MQ Explorer

Objectives

- To become knowledgeable with the concepts and navigation of MQ Explorer.

Exercise

1. Invoke the MQ Explorer and perform these tasks:

   1.1. Create a new queue manager and review the default settings.

   1.2. What is the Code Character Set ID?

   1.3. What is the maximum uncommitted message?

   1.4. What is the maximum priority?

   1.5. What is the name of the default command queue?

2. Create a new local message queue, TempQueue, with a maximum of 10 messages in the queue.

   2.1. Put messages on the queue.

   2.2. Browse the messages on the queue.

   2.3. Clear the messages on the queue.

5 Message Queue Interface and Java

There are no machine exercises for this section.

6 Connecting a Queue Manager

Objectives

- Connect a queue manager and handle exceptions.

Exercise

1. Connect to a queue manager by creating a new instance of the MQQueueManager class.

   1.1. If the connection is successful, display the maximum length and priority that this queue manager can handle.

2. Methods in the Java interface do not return a completion code and reason code.

   2.1. An exception is thrown whenever the completion code and reason code resulting from a IBM MQ call are not both zero.

   2.2. If the connection is not successful, display the completion code and reason code.

   2.3. Use the class MQException and the fields completionCode and ex.reasonCode.

7 Putting a Message

Objectives

- Connect to a queue manager and open a queue for output.
- Place several messages on the queue.

Exercise

The program will be compiled and tested with either IBM Rational Application Developer or Eclipse.

1. Code a program that will:
   1.1. Connect and open the queue that was created in the queue manager exercise.
2. Check the condition and the responses code.
   2.1. If an error occurs, display a message and the response code.
   2.2. Confirm that the operation was successful by checking the condition and response codes.
3. Write the name of all the students in the class onto the queue.
4. Close and disconnect from the queue.

8 Browsing Messages

8.1 Browse a Local Queue

Objectives

- Browse a queue displaying the content of a queue without removing the messages.

Exercise

The program will be compiled and tested with either IBM Rational Application Developer or Eclipse.

1. Code a program that will:

   1.1. Connect and open the queue that was created in the previous exercise.

   1.2. Check the condition code and responses codes.

   1.3. If an error occurs, display a message and the response code.

      1.3.1. Confirm that the operation was successful by checking the condition and response codes.

   1.4. Display every message in the queue.

      1.4.1. Do not delete any messages.

2. Close and disconnect from the queue.

8.2 Get All Messages and Delete the Retrieved Messages

Objectives

- Retrieve a queue and display the content of a queue.
- Remove the messages from the queue.

Exercise

The program will be compiled and tested with either IBM Rational Application Developer or Eclipse.

1. Code a program that will:

   1.1. Connect and open the queue that was created previously.

   1.2. Check the condition and the responses code.

   1.3. When an error occurs, display a message and response code.

      1.3.1. Confirm that the operation was successful by checking the conditions and response codes.

   1.4. Display every message in the queue and then delete all messages after it has been retrieved.

2. Close and disconnect from the queue.

8.3 Get a Message for a Specific ID

Objectives

- Search the content of a queue using the Message ID.

Exercise

The program will be compiled and tested with either IBM Rational Application Developer or Eclipse.

1. Code a program that will:

    1.1. Connect and open the queue that was created in the previous exercise.

    1.2. Check the condition and the response codes.

    1.3. If an error occurs, display a message and the response code.

        1.3.1. Confirm that the operation was successful by checking the condition code and response code.

    1.4. Delete every message in the queue.

    1.5. Write the name of each student in the class onto the queue.

        1.5.1. Store the name in the Message ID field.

    1.6. Read the queue for a specific Name - Message ID.

        1.6.1. Do not read the entire queue.

    1.7. Randomly access the selected name and if a name has been successfully found display it.

2. Close and disconnect from the queue.

8.4 Get Messages in Priority Order

Objectives

- Retrieve messages in priority order.

Exercise

The program will be compiled and tested with either IBM Rational Application Developer or Eclipse.

1. Code a program that will:

   1.1. Connect and open the queue that was created in the previous exercise.

   1.2. Check the condition and response codes.

   1.3. If an error occurs, display a message and the response code.

   1.4. Confirm that the operation was successful by checking the condition and response codes.

   1.5. Delete every message in the queue.

   1.6. Write the name of every student in the class onto the queue.

   1.7. Students with a name that starts with an A-N get a higher priority than students with names that start with O-Z.

   1.8. Browse the queue in priority order and display the names.

2. Close and disconnect from the queue.

9 Committing and Backing Out Units of Work

Objectives

- Create a unit of work and explicitly rollback a unit of work.

Exercise

The program will be compiled with the JCL provided by the instructor.

1. Code a program that will:

   1.1. Connect and open the queue that has been created previously.

   1.2. Check the condition code and responses codes.

   1.3. If an error occurs, display a message and the response code.

   1.4. Confirm that the operation was successful by checking the condition and response codes.

2. Start a unit of work.

   2.1. Write the name of each student in the class onto the queue.

   2.2. Rollback the transaction.

   2.3. Close and disconnect from the queue.

   2.4. Use a utility to browse the queue and confirm the rollback.

10 Triggers

Objectives

- Create a trigger.
- Code a program to use the trigger.

Exercise

1. Create an initiation queue.

   1.1. Create a CICS process.

   1.2. The instructor will provide the transaction ID.

2. Create a trigger which executes a process every time a message is placed into a queue.

   2.1. The triggered process will store the delivered message to a queue.