Course Exercises Guide

IBM MQ V9 System Administration (using Linux for labs)

Course code WM154 ERC 1.0

IBM Training

April 2017 edition

Contents

Trademarks

Exercises description

This course includes the following exercises:

- Using commands to create queue manager and queues
- Using IBM MQ Explorer to create queue managers and queues
- Using IBM MQ sample programs to test the configuration
- Connecting queue managers
- Connecting an IBM MQ client
- Implementing a trigger monitor
- Running an IBM MQ trace
- Controlling access to IBM MQ
- Using a media image to restore a queue
- Back up and restore IBM MQ object definitions
- Implementing a basic cluster
- Monitoring IBM MQ for performance

In the exercise instructions, you can check off the line before each step as you complete it to track your progress.

Most exercises include required sections, which should always be completed. It might be necessary to complete these sections before you can start later exercises. If you have sufficient time and want an extra challenge, some exercises might also include optional sections that you can complete.

The lab image for this course is based on Ubuntu 14.4. For more information about Ubuntu, see the Ubuntu desktop guide: https://help.ubuntu.com/lts/ubuntu-help/index.html

The primary user ID and password for the lab image for this course are: `localuser`, `passw0rd`

Exercise 1. Using commands to create a queue manager and queues

Estimated time

00:30

Overview

In this exercise, you use IBM MQ commands to create a queue manager, start it, and then create queues. You also create and run an IBM MQ script command file.

Objectives

After completing this exercise, you should be able to:

- Use IBM MQ commands to create a local queue manager, local queues, and alias queues
- Use IBM MQ commands to display and alter queue manager and queue attributes
- Create and run an IBM MQ command file

Introduction

In this exercise, you create and start a queue manager. You then create local queues and alter queue and queue manager attributes.

You also create a script file of IBM MQ commands and run the command file from a command.

Finally, you create alias queues and alter the queue attributes.

Requirements

- IBM MQ is installed on the system
- A text editor

Important

The exercises in this course use a set of lab files that might include scripts, applications, files, solution files, PI files, and others. The course lab files can be found in the `C:/labfiles` directory. The exercises point you to the lab files as you need them.

Exercise instructions

Part 1: Create and start a queue manager

__ 1. Log on to the VMware image or system.

W **Windows**

Log in as `Administrator` with a password of `passw0rd`

L **Linux**

Log in as `localuser` with a password of `passw0rd`

__ 2. Open a command (terminal) window.

__ 3. Create a queue manager that is named: **`QM01`**
Use this queue manager for all steps in this exercise.

Note: The queue manager and dead-letter queue names are case-sensitive.

W **Windows**

Type: **`crtmqm -u SYSTEM.DEAD.LETTER.QUEUE QM01`**

L **Linux**

Create the queue manager with user-based authorizations that are enabled by adding the **`-oa user`** option to the create queue manager command. Type:

```
crtmqm -oa user -u SYSTEM.DEAD.LETTER.QUEUE QM01
```

This queue manager (QM01) is used in a security exercise later in this course (Exercise 8) that assumes that user-based authorizations are enabled on the Linux queue manager. This option and topic are described in more detail in Unit 10.

You should see a message that the queue manager and a queue manager directory were created successfully.

__ 4. Start the queue manager.

Type: **`strmqm QM01`**

You should see a message that the queue manager QM01 started by using V9.0.0.0.

Part 2: Use the IBM MQ MQSC commands interactively

In this part of the exercise, you enter all the IBM MQ commands interactively and display the results in the command window.

__ 1. Use the control command **`runmqsc`** to start MQSC mode and complete the following tasks.

Type: **`runmqsc QM01`**

You see message that the MQSC for the queue manager QM01 is starting. You are in MQSC mode until you enter **`END`**. No command prompt is displayed in MQSC mode.

__ a. Display all the attributes of the queue manager.

Type: **`DISPLAY QMGR`**

Note: **`QMGR`** is a literal string and is not replaced with your queue manager name.

__ b. Verify that the DEADQ attribute is set to: `SYSTEM.DEAD.LETTER.QUEUE`

__ c. Display all the queues with queue names that begin with the characters **`SYSTEM`**.

Type: **`DISPLAY Q(SYSTEM*)`**

__ d. Create a local queue that is named **QL.A** with a text description of “QL.A TEXT”.

Type: **`DEFINE QL(QL.A) REPLACE DESCR('QL.A TEXT')`**

__ e. Display all the attributes of the local queue that is name QL.A and verify that the text that is entered in the previous step is the value for the DESC attribute.

Type: **`DISPLAY QL(QL.A)`**

__ f. Change the maximum number of messages that are allowed on the local queue QL.A (MAXDEPTH) from 5000 messages to 1000 messages.

Type: **`ALTER QL(QL.A) MAXDEPTH(1000)`**

__ g. Display the queue attributes for QL.A and verify the change to the MAXDEPTH attribute.

Type: **`DISPLAY QL(QL.A)`**

Note

You can use the Up arrow key in the command window to display and rerun a previous command.

Verify that the `ALTER` command changed only the attribute that you specified on the command. You should see that the MAXDEPTH attribute was modified and that the DESC attribute is unchanged.

__ h. Define another local queue that is named **QL.B** with a text description.

Type: **`DEFINE QL(QL.B) REPLACE DESCR('QL.B Text')`**

__ i. Display all the attributes of the queue and verify that the text appears in the DESC attribute.

Type: **`DISPLAY QL(QL.B)`**

__ j. Change the maximum number of messages that are allowed on the queue to 2000 by using the `DEFINE` command with `REPLACE` instead of the `ALTER` command.

Type: **`DEFINE QL(QL.B) REPLACE MAXDEPTH(2000)`**

__ k. Display the queue attributes of QL.B.

Type: **`DISPLAY QL(QL.B)`**

The `DEFINE` command with `REPLACE` changes the attributes that you specify on the command and sets the other attributes to their initial values. You should see that the `MAXDEPTH` attribute was modified and that the `DESC` attribute is now blank.

__ 2. Exit the **`runmqsc`** mode.

Type: **`END`**

Part 3: Use the IBM MQ MQSC commands in a command file

__ 1. Using a text editor such as Notepad on Windows or gedit on Linux, prepare an IBM MQ command file with the same commands that you entered interactively in Part 2 of this exercise. Name the file **`CreateQs.mqsc`** and save it in your lab files directory.

W **Windows**

The lab files directory is: **`C:\labfiles`**

L **Linux**

The lab files directory is: **`/home/localuser/labfiles`**

In the text file, enter the following commands and then save the file.

```
DIS QMGR
DIS Q(SYSTEM*)
DEF QL(QL.A) REPLACE DESCR('QL.A Text')
DIS QL(QL.A)
ALTER QL(QL.A) MAXDEPTH(1000)
DIS QL(QL.A)
DEF QL(QL.B) REPLACE DESCR('QL.B Text')
DEF QL(QL.B) REPLACE MAXDEPTH(2000)
DIS QL(QL.B)
```

__ 2. Process the command file and direct the results to a file that is named **`report.txt`**.

__ a. In the command window, change directories to the lab files directory that contains the MQSC file that you created in Step 1.

__ b. Run the command file and redirect the output to `report.txt`.

Type: **`runmqsc QM01 < CreateQs.mqsc > report.txt`**

__ 3. Using a text editor, open the file that is named **`report.txt`** and verify that each command was successful.

Part 4: Work with alias queues and queue attributes

In this part of the exercise, you create alias queues and modify queue attributes. You use these queues in Exercise 3.

__ 1. Start MQSC mode for the queue manager QM01.

Type: **`runmqsc QM01`**

__ 2. Verify the local queues QL.A and QL.B exist.

Type: **`DIS QL(QL*)`**

__ 3. Create an alias queue on QM01 that is named **QA.A** that resolves to the local queue QL.A.

Type: **`DEF QA(QA.A) TARGET(QL.A)`**

__ 4. Display the attributes of the alias queue QA.A and verify that queue TYPE is **QALIAS** and that TARGET is **QL.A**.

Type: **`DIS QA(QA.A)`**

__ 5. Modify the alias queue QA.A to inhibit PUT requests.

Type: **`ALTER QA(QA.A) PUT(DISABLED)`**

__ 6. Display the attributes of the alias queue QA.A and verify that queue PUT attribute is now set to **DISABLED**.

__ 7. Modify the local queue QL.B to inhibit PUT requests.

Type: **`ALTER QL(QL.B) PUT(DISABLED)`**

__ 8. Display the attributes of the local queue QL.B and verify that queue PUT attribute is now set to **DISABLED**.

__ 9. Create an alias queue that is named **QA.B** that resolves to the local queue QL.B.

Type: **`DEF QA(QA.B) TARGET(QL.B)`**

__ 10. Display the attributes of the alias queue QA.B and verify that queue TYPE is **QALIAS** and that TARGET is **QL.B**.

__ 11. Exit **`runmqsc`** mode.

Type: **`END`**

End of exercise

Exercise review and wrap-up

You should now be able to:

- Create and start a queue manager
- Use control commands and MQSC command files to create and modify local queues
- Display attributes of IBM MQ objects
- Change queue attributes and create alias queues

Exercise 2. Using IBM MQ Explorer to create queue managers and queues

Estimated time

00:30

Overview

In this exercise, you use IBM MQ Explorer to create a queue manager, start it, and then create queues. You also use IBM MQ Explorer to create queue manager sets to simplify the management of many queue managers.

Objectives

After completing this exercise, you should be able to:

- Use IBM MQ Explorer to create a local queue manager, local queues, and alias queues
- Use IBM MQ Explorer to display and modify queue manager and queue properties
- Use IBM MQ Explorer to create a queue manager set

Introduction

In this exercise, you use IBM MQ Explorer to create a queue manager. Then, you create local queues and modify queue properties. Finally, you create queue manager sets to simplify the management of multiple queue managers.

Requirements

- IBM MQ and IBM MQ Explorer are installed on the system
- A text editor

Exercise instructions

Part 1: Create a queue manager

__ 1. Start IBM MQ Explorer.

Windows

From the Windows **Start** menu, click **MQ Explorer (Installation 1)** or click the IBM MQ Explorer icon in the taskbar.

Linux

In a terminal window, type: `MQExplorer`

When you start IBM MQ Explorer by using this command, it runs in the foreground. You must keep this terminal window open while you are using IBM MQ Explorer.

When it is necessary to open a new terminal window for running sample programs, for example, right-click and then click **New Terminal**.

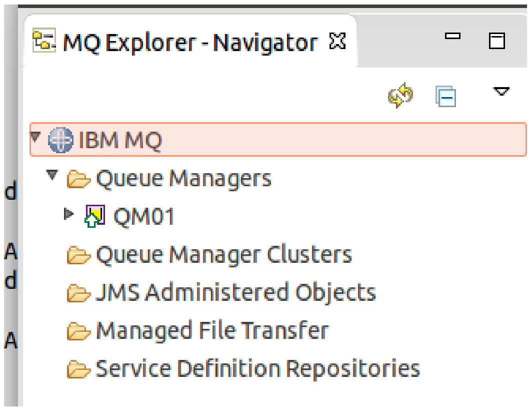
__ 2. IBM MQ Explorer automatically recognizes and manages any queue managers that are defined on the same system.

If you completed Exercise 1, verify that IBM MQ Explorer found the queue manager QM01.

__ a. In the **MQ Explorer - Navigator** view, expand **IBM MQ**.

__ b. Expand the **Queue Managers** folder.

__ c. Verify that the queue manager QM01 is listed under the **Queue Managers** folder.



__ 3. Create a queue manager that is named **QM02** that uses a dead-letter queue that is named **QM02.DLQ** and listens on port **1416**. Use this queue manager for all the remaining steps in this exercise.

__ a. In the **MQ Explorer - Navigator** view, right-click **Queue Managers** and then click **New > Queue Manager**.

__ b. For the **Queue Manager name**, type: **`QM02`**
For the **Dead-letter queue**, type: **`QM02.DLQ`**
Click **Next**.

__ c. Click **Next** to accept the default values for logging.

__ d. Click **Next** to accept the default values for the configuration options.

__ e. On the listener options page, enter **1416** for the **Listen on port number**. Click **Finish**.

__ 4. Verify that the queue manager **QM02** now appears under the **Queue Managers** folder in the **MQ Explorer - Navigator** view.

__ 5. Verify that the queue manager QM02 is running.

The status icon next to queue name should contain an up pointing green arrow. You can also click the queue manager QM02 in the **Queue Managers** folder to display and verify the queue manager status.

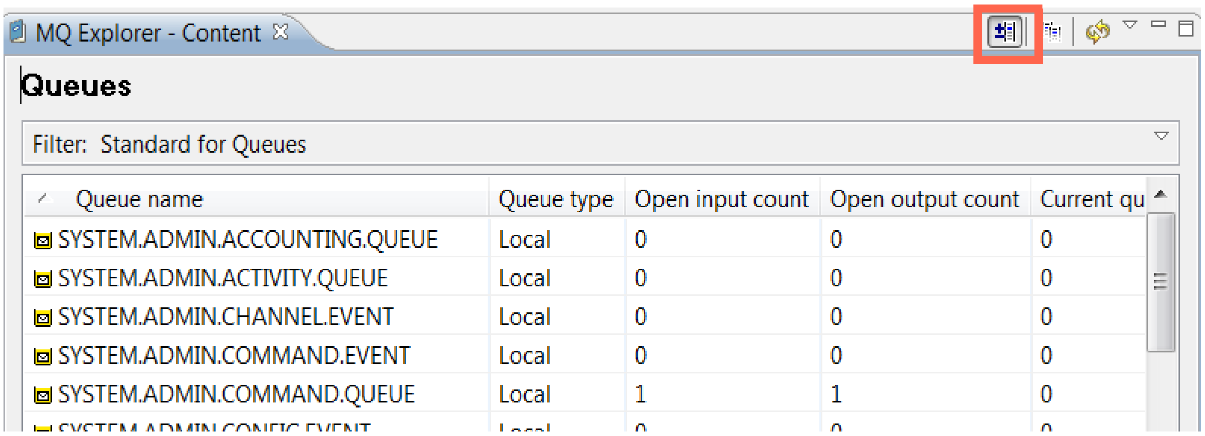
Part 2: Create and modify local queues

__ 1. Display the SYSTEM queues for QM02.

__ a. Expand **Queue Managers > QM02** in the **MQ Explorer - Navigator** view.

__ b. Click **Queues** under **QM02**.

__ c. In the **Queues** content view, click the **Show System Objects** icon.



Queues

Filter: Standard for Queues

Queue name	Queue type	Open input count	Open output count	Current qu
SYSTEM.ADMIN.ACCOUNTING.QUEUE	Local	0	0	0
SYSTEM.ADMIN.ACTIVITY.QUEUE	Local	0	0	0
SYSTEM.ADMIN.CHANNEL.EVENT	Local	0	0	0
SYSTEM.ADMIN.COMMAND.EVENT	Local	0	0	0
SYSTEM.ADMIN.COMMAND.QUEUE	Local	1	1	0

__ 2. The **Show System Queues** icon is a toggle. In the **Queues** view, click the **Show System Objects** icon again to hide the SYSTEM queues.

__ 3. Create a local queue that is named **QM02.DLQ**. You identified this queue as the dead-letter queue for QM02 when you created the queue manager.

__ a. Right-click **Queues** under **QM02** in the **MQ Explorer - Navigator** view and then click **New > Local Queue**.

__ b. For the queue **Name**, type **`QM02.DLQ`** and then click **Finish**.

__ c. If a confirmation window is displayed, click **OK**.

__ d. Verify that the queue now appears in the **Queues** content view.

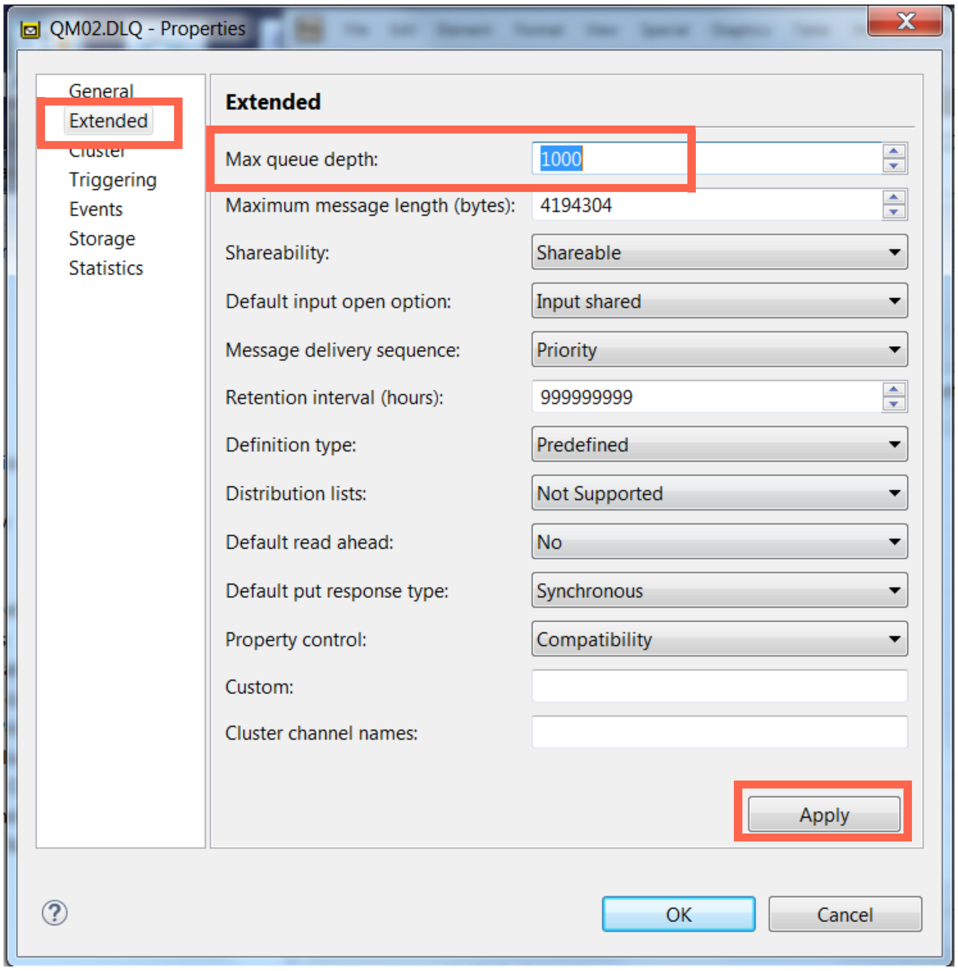
__ 4. Change the maximum number of messages that are allowed on the queue QM02.DLQ to 1000.

__ a. In the **Queues** view for QM02, right-click **QM02.DLQ** and then click **Properties**.

__ b. Click **Extended**.

__ c. Change the **Max queue depth** property value to **`1000`**.

__ d. Click **Apply** and then click **OK**.



Max queue depth: 1000
Maximum message length (bytes): 4194304
Shareability: Shareable
Default input open option: Input shared
Message delivery sequence: Priority
Retention interval (hours): 999999999
Definition type: Predefined
Distribution lists: Not Supported
Default read ahead: No
Default put response type: Synchronous
Property control: Compatibility
Custom:
Cluster channel names:

__ 5. Create another local queue on QM02 that is named **QL.B**.

__ a. Right-click **Queues** under QM02 in the **MQ Explorer - Navigator** view and then click **New > Local Queue**.

__ b. For the queue **Name**, type **`QL.B`** and then click **Finish**.

__ c. If a confirmation window is displayed, click **OK**.

__ d. Verify that queue QL.B is now listed in the **Queues** content view for QM02.

__ 6. Change the default persistence on the local queue QL.B to **Persistent**.

__ a. In the **Queues** view for QM02, right-click **QL.B** and then click **Properties**.

__ b. On the **General** page, change the **Default persistence** property to **Persistent**.

__ c. Click **Apply** and then click **OK**.

__ d. In the **Queues** view, scroll to the right and verify that the **Default persistence** column now shows **Persistent** for QL.B.

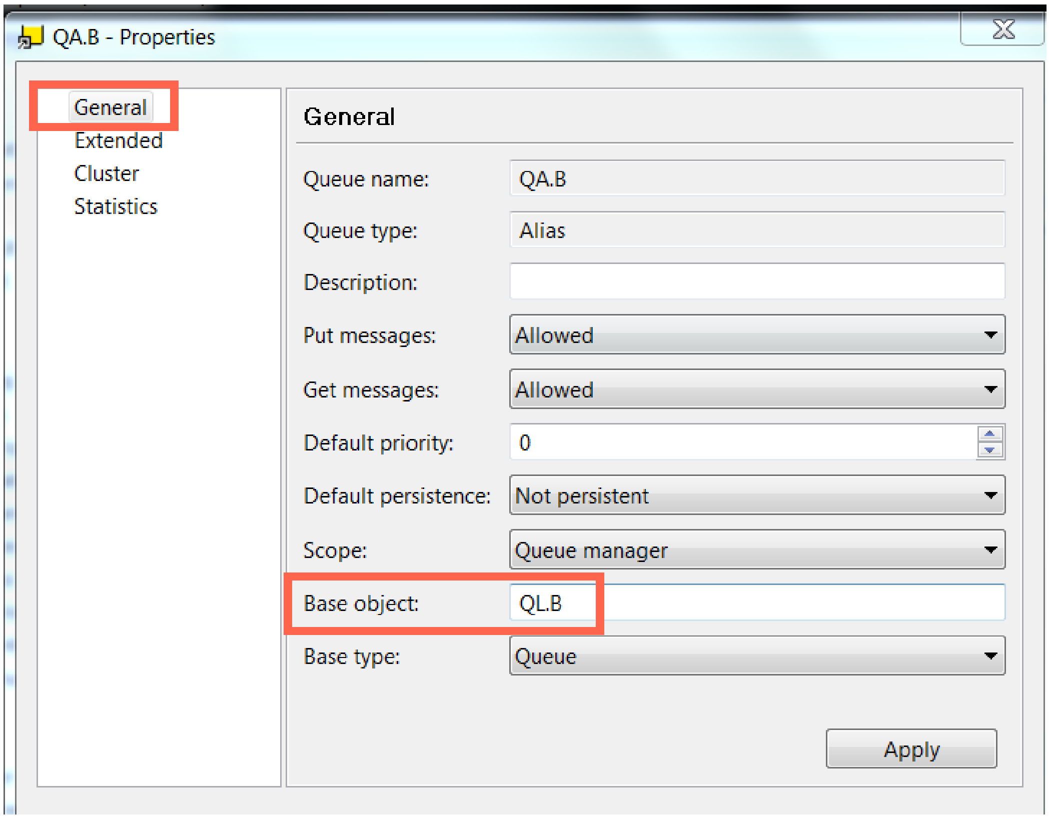
__ 7. Create an alias queue on QM02 that is named **QA.B** that resolves to the local queue QL.B.

__ a. Right-click **Queues** under QM02 in the **MQ Explorer - Navigator** view and then click **New > Alias Queue**.

__ b. For the queue **Name**, type `QA.B` and then click **Next**.

__ c. On the **General** page, type `QL.B` for the **Base object** property and then click **Finish**.

__ d. If a confirmation window is displayed, click **OK**.



General

Queue name: QA.B
Queue type: Alias
Description:
Put messages: Allowed
Get messages: Allowed
Default priority: 0
Default persistence: Not persistent
Scope: Queue manager
Base object: QL.B
Base type: Queue

__ 8. Inhibit PUT requests on the QA.B alias queue on QM02.

__ a. In the **Queues** view for QM02, right-click **QA.B** and then click **Properties**.

__ b. On the **General** page, change the **Put messages** property to **Inhibited**.

__ c. Click **Apply** and then click **OK**.

__ d. Verify that the **Put messages** column in the **Queues** view now contains **Inhibited** for the queue QA.B.

Part 3: Create queue manager sets

In this part of the exercise, you create a queue manager set that is named **Production**. You then create a custom filter that uses the value in the queue manager **Description** field to identify "Production" queue managers to include in the new set.

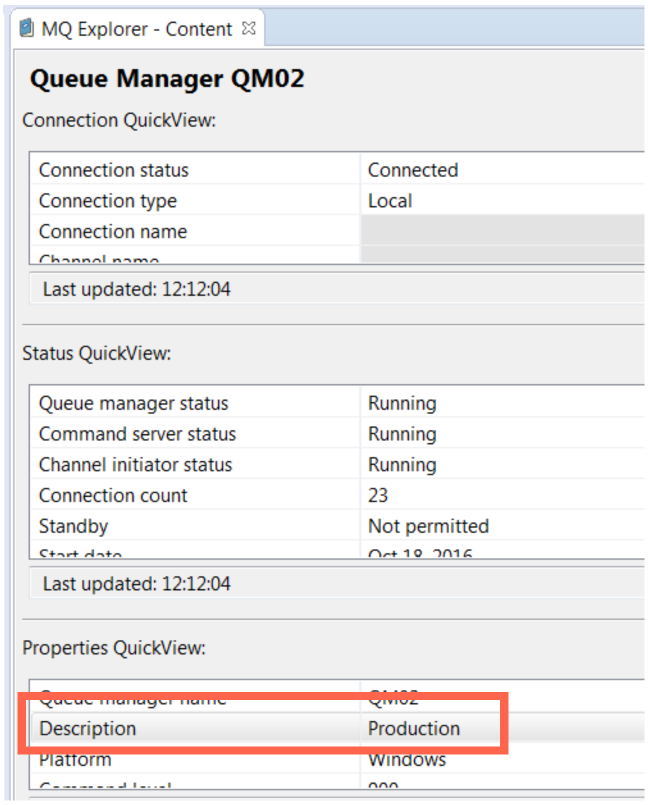
__ 1. Change the description of the QM02 queue manager to **Production**.

__ a. In the **MQ Explorer - Navigator** view, right-click **QM02** and then click **Properties**.

__ b. On the **General** properties page, type `Production` in the **Description** field.

__ c. Click **Apply** and then click **OK**.

__ d. In the **Queue Manager QM02** view, verify that the **Description** property is now set to **Production**.


MQ Explorer - Content

Queue Manager QM02

Connection QuickView:

Connection status	Connected
Connection type	Local
Connection name	
Channel name	

Last updated: 12:12:04

Status QuickView:

Queue manager status	Running
Command server status	Running
Channel initiator status	Running
Connection count	23
Standby	Not permitted
Start date	Oct 18, 2016

Last updated: 12:12:04

Properties QuickView:

Queue manager name	QM02
Description	Production
Platform	Windows
Command level	900

__ 2. Create a queue manager set that is named **Production** and automatically add any queue managers with the **Description** property set to `Production` to the queue manager set.

__ a. Right-click **Queue Managers** in the **MQ Explorer-Navigator** view, and then click **Sets > New Set**.

__ b. For the set **Name**, type `Production` and then select **Automatic** for the set type. Click **Next**.

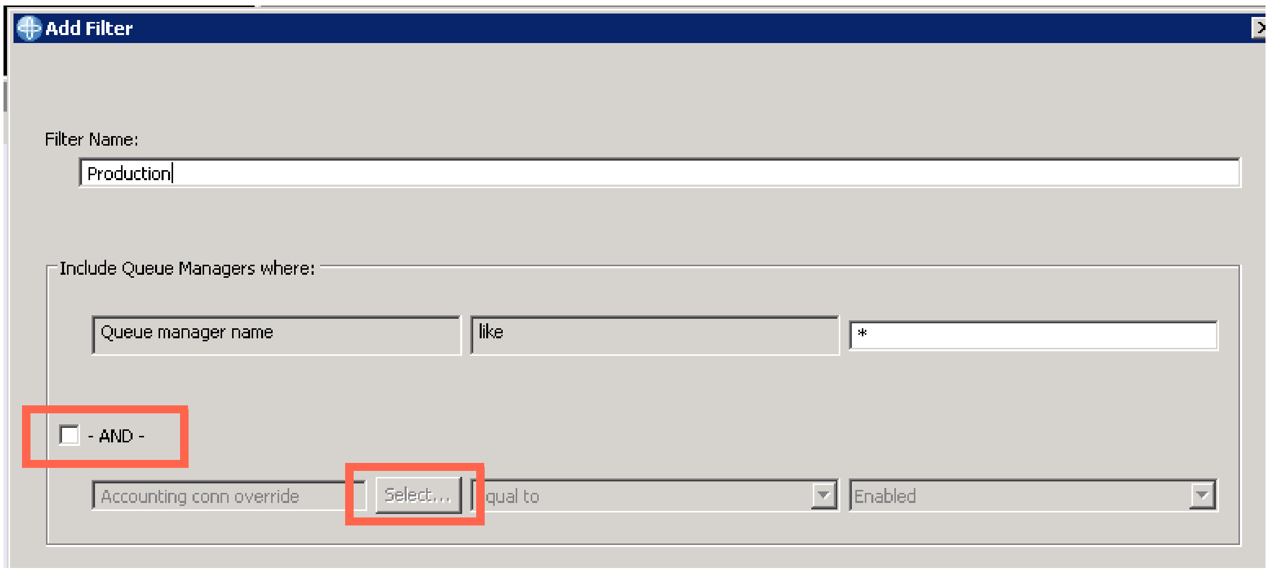
__ c. Click **Manage Filters** to define a new filter that references the queue manager **Description** property.

__ d. Click **Add** in the **Manage Filters** window.

__ e. In the **Add Filter** window, type `Production` for the **Filter Name**.

__ f. The first part of the filter currently evaluates all queue managers. This filter should evaluate all queue managers, so no changes are required to the first filter evaluation.

__ g. Add an AND clause to the filter that tests for the text `Production` in the queue manager **Description** property. Select the **AND** check box, and then click **Select**.


Add Filter

Filter Name: Production

Include Queue Managers where:

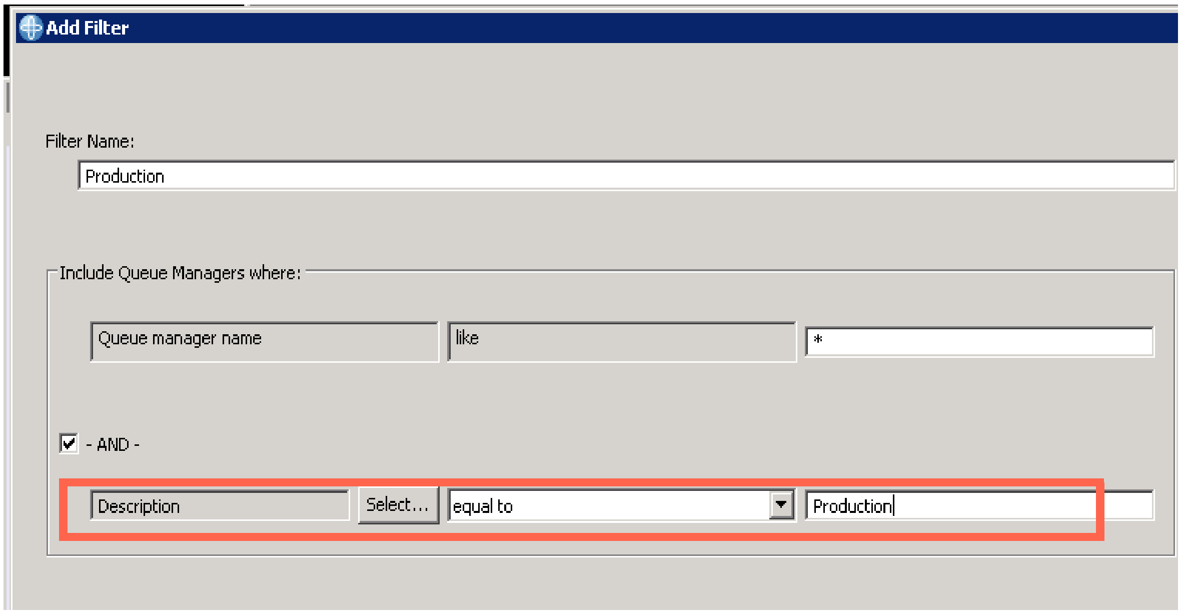
Queue manager name | like | *

☐ - AND -

Accounting conn override | Select... | equal to | Enabled

__ h. In the **Select Attribute** window, scroll down, select the **Description** attribute, and then click **OK**.

__ i. Complete the AND statement in the filter definition by setting the evaluation option to **equal to** and entering: `Production`


Add Filter

Filter Name: Production

Include Queue Managers where:

Queue manager name | like | *

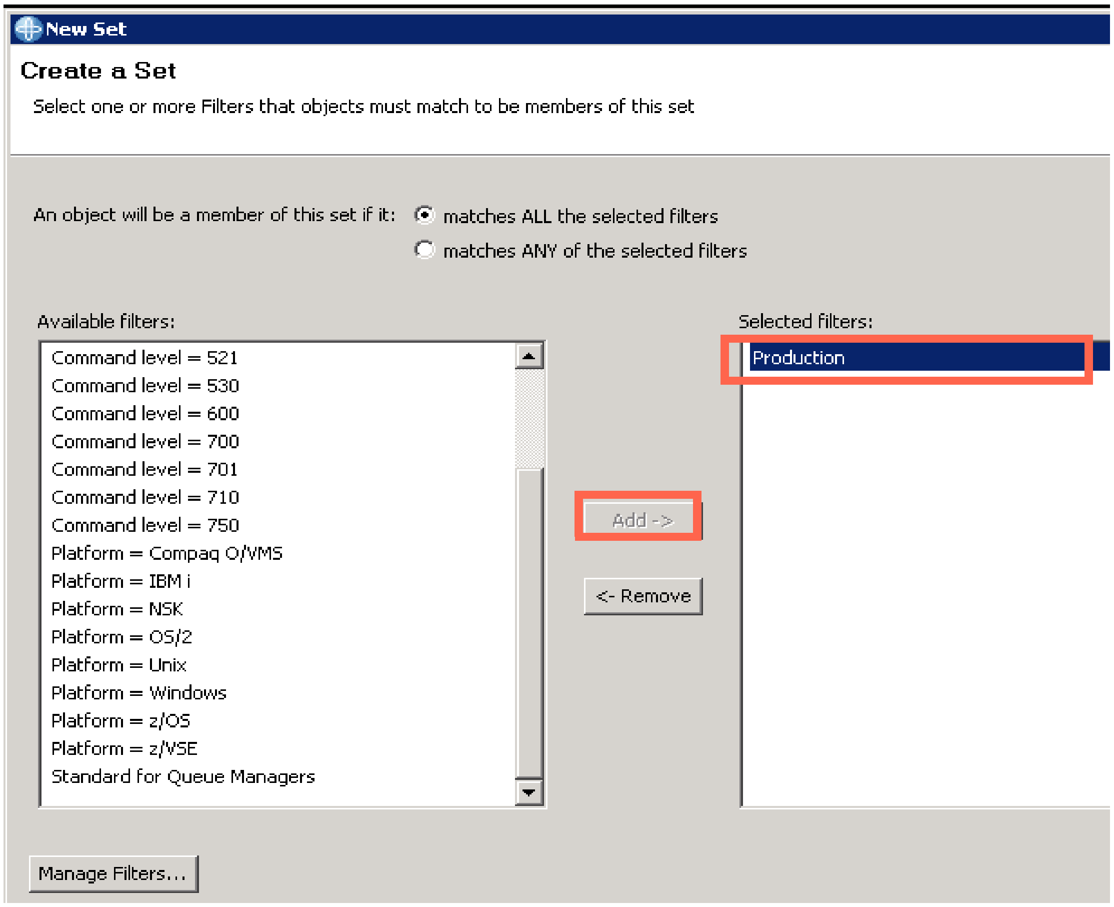
☑ - AND -

Description | Select... | equal to | Production

__ j. Click **OK** on the **Add Filter** window.

__ k. Click **OK** in the **Manage Filters** window.

__ l. In the **New Set** window, scroll down the **Available filters** list and then select **Production** from the list of available filters. Click **Add** to move **Production** to the **Selected filters** pane.


New Set

Create a Set

Select one or more Filters that objects must match to be members of this set

An object will be a member of this set if it: [✓ matches ALL the selected filters] matches ANY of the selected filters

Available filters:

Command level = 521
Command level = 530
Command level = 600
Command level = 700
Command level = 701
Command level = 710
Command level = 750
Platform = Compaq O/VMS
Platform = IBM i
Platform = NSK
Platform = OS/2
Platform = Unix
Platform = Windows
Platform = z/OS
Platform = z/VSE
Standard for Queue Managers

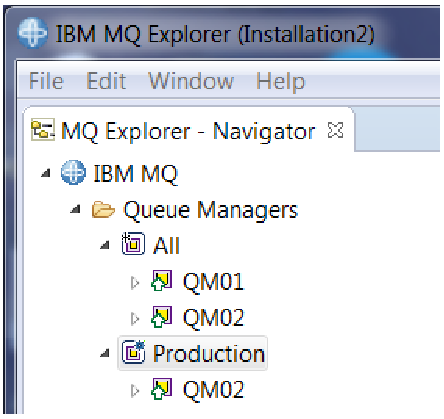
Selected filters:

Production

__ m. Click **Finish**.

__ 3. Verify that the **Queue Managers** folder in the **MQ Explorer - Navigator** view now contains two sets: **All** and **Production**.

__ 4. Verify that the queue manager QM02 is a member of the **Production** set.



Exercise cleanup

By using MQ Explorer or MQSC, stop the queue manager QM02.

- To stop the queue manager by using MQ Explorer, right-click the queue manager in the **MQ Explorer - Navigator** view and then click **Stop > Controlled**.
- To stop the queue manager by using a command, type: **`endmqm -c QM01`**

End of exercise

Exercise review and wrap-up

You should now be able to use IBM MQ Explorer to:

- Create a queue manager and modify queue manager properties
- Create local and alias queues and modify queue properties
- Create a queue manager set that uses a custom filter to automatically assign it to the set

Exercise 3. Using IBM MQ sample programs to test the configuration

Estimated time

00:30

Overview

In this exercise, you use the IBM MQ sample programs to put, get, and browse queues and test the queue manager configuration. You then use IBM MQ Explorer and IBM MQ commands to verify the actions of the sample programs. You also define and test an alias queue.

Objectives

After completing this exercise, you should be able to:

- Use IBM MQ sample programs to put messages onto a queue, browse messages on a queue, and get messages from a queue
- Use IBM MQ Explorer and IBM MQ commands to display queue contents
- Define and test an alias queue that refers to another queue

Introduction

In this exercise, you test the queue manager QM01 and queues that you defined in Exercise 1 by using the MQ sample programs and MQ Explorer to put, get, and browse messages.

Linux

The sample programs are in the **`/opt/mqm/samp/bin`** subdirectory. The path to this directory is included in the **`/etc/environment`** file on the lab image for this course.

Windows

The sample programs are in **`C:\Program Files\IBM\MQ\Tools\c\Samples\Bin64`**. The PATH statement is already set on the Windows lab image.

The IBM MQ sample programs are run in a command window. The **`amqsput`**, **`amqsget`**, and **`amqsbcg`** sample programs that are used in this exercise accept two parameters. The first parameter is required and is the name of a queue. The second parameter is optional and is the name of a queue manager. If the second parameter is omitted, the default queue manager is assumed. Both parameters are case-sensitive.

A brief description of each program is provided.

- The **`amqsput`** program connects to the queue manager and opens the queue. It reads lines of text from the standard input device, generates a message from each line of text, and puts the messages on the named queue. After the program reads a null line or the EOF character from the standard input device, it closes the queue, and disconnects from the queue manager.
  The command to start the program is: `amqsput` *`QName QmgrName`*

  Example: **`amqsput Q1 QM01`**

- The **`amqsget`** program connects to the queue manager, opens the queue for input, and gets all the messages from the queue. It writes the text within the message to the standard output device, waits 15 seconds (60 seconds if a message does not exist at the start) in case any more messages are put on the queue. The program then closes the queue and disconnects from the queue manager.
  The command to start the program is: `amqsget` *`QName QmgrName`*

  Example: **`amqsget Q2 QM01`**

- The **amqsbcg** program connects to the queue manager and opens the queue for browsing. It browses all the messages on the queue and writes their contents, in both hexadecimal and character format, to the standard output device. It also returns, in a readable format, the fields in the message descriptor for each message. It then closes the queue and disconnects from the queue manager.
  The command to start the program is: `amqsbcg` *`QName QmgrName`*

  Example: **`amqsbcg Q3 QM01`**

Requirements

- IBM MQ and IBM MQ Explorer
- Queue manager QM01 and queues that were created in Exercise 1
- The IBM MQ sample programs **`amqsput`**, **amqsbcg**, and **`amqsget`**
- A text editor

Exercise instructions

Part 1: Make QM01 is the default queue manager

The default queue manager processes any commands for which a queue manager name is not explicitly specified. In this part of the exercise, you make QM01 the default queue manager so that you do not have to specify the queue manager name when you run the IBM MQ sample programs.

__ 1. Open the MQ Explorer unless it is already open.

__ 2. Right-click **IBM MQ**, and then select **Properties**. The properties for MQ are displayed.

__ 3. In the **Default queue manager** name, type: **`QM01`**.

__ 4. Click **Apply** and then click **OK**.

Part 2: Put, get, and browse messages

In this part of the exercise, you use the queues that you created in Exercise 1 and the IBM MQ sample programs to put, get, and browse messages.

__ 1. Use a sample program to put two messages on the local queue QL.A on QM01.

  __ a. In a terminal window, type: **`amqsput QL.A`**

  The program responds with:

```
Sample AMQSPUT0 start
target queue is QL.A
```

  __ b. Enter some data and then press Enter. Each line of data that is entered becomes the data portion of a new MQ message.

  __ c. To end the sample program, press Enter on a blank line.

Troubleshooting

If the program does not run successfully, an MQ reason code is returned.

To interpret the reason code, use the control command `mqrc` at the **`C:`** or **`$`** prompt followed by the four-digit reason code. For example, type: `mqrc 2085`

__ 2. Use MQ Explorer to display the queue contents and browse the messages.

  __ a. Select the **Queue** folder under QM01 in the **MQ Explorer - Navigator** view to display the **Queues** view.

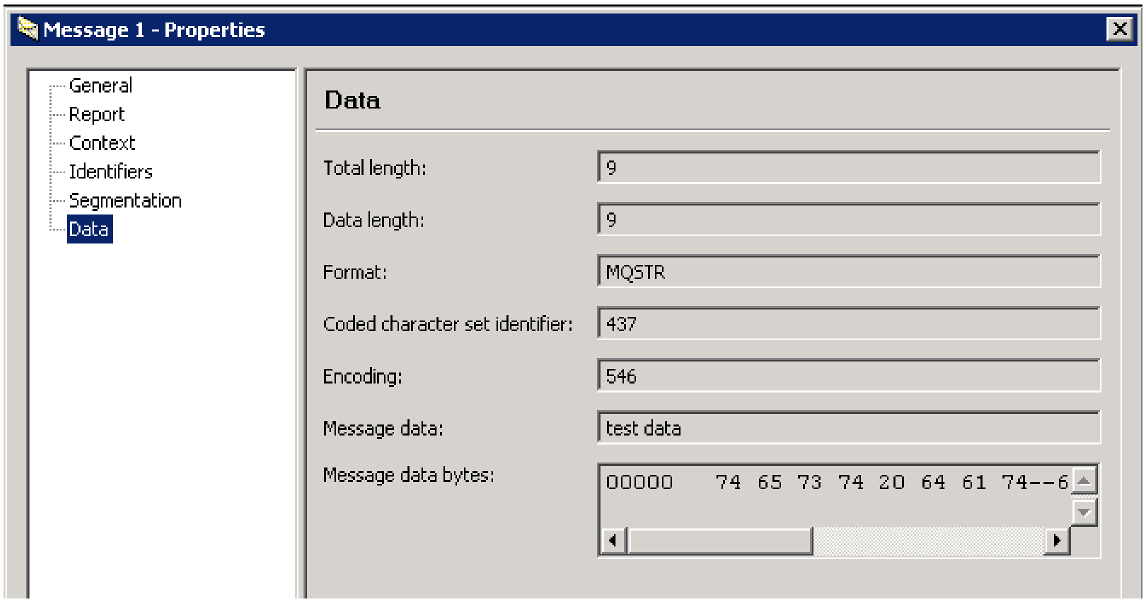
  __ b. Right-click **QL.A** in the **Queues** view for QM01 and then click **Browse messages**. The **Message browser** list contains one message for every message that you entered with the **`amqsput`** sample program.

  __ c. Double-click a message from the list of messages to display its properties.

  __ d. Click the **Data** properties tab to view the message data.

  __ e. Click **Close** to close the **Properties** window.

__ f. Click **Close** to close the message browser window.



Data

Total length: 9

Data length: 9

Format: MQSTR

Coded character set identifier: 437

Encoding: 546

Message data: test data

Message data bytes:

```
00000    74 65 73 74 20 64 61 74--6
```

__ 3. Use a sample program to browse the messages on the queue QL.A. Direct the command results to a file and view the results.

In the terminal window, type: **`amqsbcg QL.A > out.txt`**

If the program does not run successfully, can you determine the reason for the failure?

__ 4. Use a sample program to get the messages from the queue and empty the queue.

Type: **`amqsget QL.A`**

The **`amqsget`** sample program can take a few seconds to run. It is finished when you see the message:

```
no more messages
Sample AMQGET0 end
```

If the program does not run successfully, can you determine the reason for the failure?

__ 5. Use the **`amqsput`** sample program to put three messages on the queue QL.A.

__ 6. Use an MSQC command to show the current depth (CURDEPTH) of QL.A (the number of messages on the queue) and verify that the messages are on the queue.

__ a. Open a new terminal window.

__ b. Type: **`runmqsc QM01`**

__ c. Type: **`DIS Q(QL.A) CURDEPTH`**

__ 7. Define a new local queue on QM01 that is named **`QL.X`** with the attributes of the local queue QL.A.

In the MQSC window, type: **`DEF QL(QL.X) LIKE(QL.A)`**

__ 8. Use a sample program or MQ Explorer to put some messages on QL.X.

To use the sample program, type **`amqsput QL.X`** and then enter some messages.

To use MQ Explorer, right-click QL.X in the **Queues** view and then click **Put Test Message**.

__ 9. Clear the messages from the local queue QL.A.

In the window that is running MQSC, type: **`CLEAR QL(QL.A)`**

__ 10. Try to clear the messages from the alias queue QA.A.

In the MQSC window, type: **`CLEAR QA(QA.A)`**

You should receive a syntax error that indicates that the CLEAR command is valid for local queues and topic strings only and not alias queues (QALIAS).

You cannot clear messages from an alias queue because it is a pointer to another queue and does not hold messages.

__ 11. Delete the local queue QL.X.

In the MQSC window, type: **`DELETE QL(QL.X)`**

You should see that you cannot delete the queue because it is not empty.

To delete the queue and the messages, type:

```
DELETE QL(QL.X) PURGE
```

Part 3: Work with alias queues

In this part of the exercise, you use the alias queues that you created on QM01 in Exercise 1.

In Exercise 1, you created an alias queue that is named QA.A that points to the local queue QL.A and an alias queue that is named QA.B that points to local queue QL.B.

__ 1. Using MQ Explorer or MQSC, determine whether the **PUT messages** attribute is set to **allowed** (enabled) or **inhibited** (disabled) for the alias queue QA.A.

To use MQ Explorer, click the **Queues** folder under QM01 to display the **Queues** view. The **Put messages** column shows the current state of the PUT attribute.

To use MQSC to display the PUT attribute, type: **`DIS QA(QA.A)`**

__ 2. If the PUT requests on the alias queue QA.A are allowed (enabled), change the queue PUT attribute to inhibit them.

In window that is running MQSC, type: **`ALTER QA(QA.A) PUT(DISABLED)`**

__ 3. Use MQ Explorer or MQSC to determine whether the **PUT messages** attribute is set to **allowed** (enabled) or **inhibited** (disabled) for the local queue QL.B.

If the PUT requests on the local queue QL.B are allowed, change the queue PUT attribute to inhibit them.

In the window that is running MQSC, type: **`ALTER QL(QL.B) PUT(DISABLED)`**

__ 4. Use the **`amqsput`** sample program to try to put messages on both the alias and local queues.

Run the sample program in the command window.

- For QL.A, type: **`amqsput QL.A`**
- For QA.A, type: **`amqsput QA.A`**
- For QL.B, type: **`amqsput QL.B`**
- For QA.B, type: **`amqsput QA.B`**

The PUT request for QL.A succeeds but the other requests fail with reason code 2051 because PUT is inhibited.

QA.A and QL.B fail because PUT is inhibited on those queues.

QA.B fails because it is an alias queue for the local queue QL.B, which has PUT inhibited.

Exercise cleanup

__ 1. Change the local queue QL.B to allow PUT requests.

In the MQSC window, type: **`ALTER QL(QL.B) PUT(ENABLED)`**

__ 2. Change the alias queue QA.A to allow PUT requests.

In the MQSC window, type: **`ALTER QA(QA.A) PUT(ENABLED)`**

__ 3. Use MQ Explorer to clear any messages on the local queues QL.A and QL.B.

- __ a. In the **Queues** view, right-click QL.A and then click **Clear Messages**.
- __ b. Select **Clear will be used using CLEAR command** and then click **CLEAR**.
- __ c. Click **OK** in the verification window.
- __ d. Verify that the **Current queue depth** column value for QL.A is zero.
- __ e. If any messages are on QL.B, repeat steps a – d to clear QL.B.

Note

You must use MQGET instead of CLEAR to empty a queue if:

- Uncommitted messages are on the queue and they were put on the queue under sync point
- An application currently has the queue open

End of exercise

Exercise review and wrap-up

You should now be able to:

- Use MQ sample programs to put messages onto a queue, browse messages on a queue, and get messages from a queue
- Use MQ Explorer and MQSC to display queue contents
- Use MQSC to change a queue to inhibit or allow PUTs
- Use MQ Explorer and MQSC clear and delete queues

Exercise 4. Connecting queue managers

Estimated time

01:00

Overview

In this exercise, you create channels between two queue managers. You use the IBM MQ sample programs to test the connection between the queue managers.

Objectives

After completing this exercise, you should be able to:

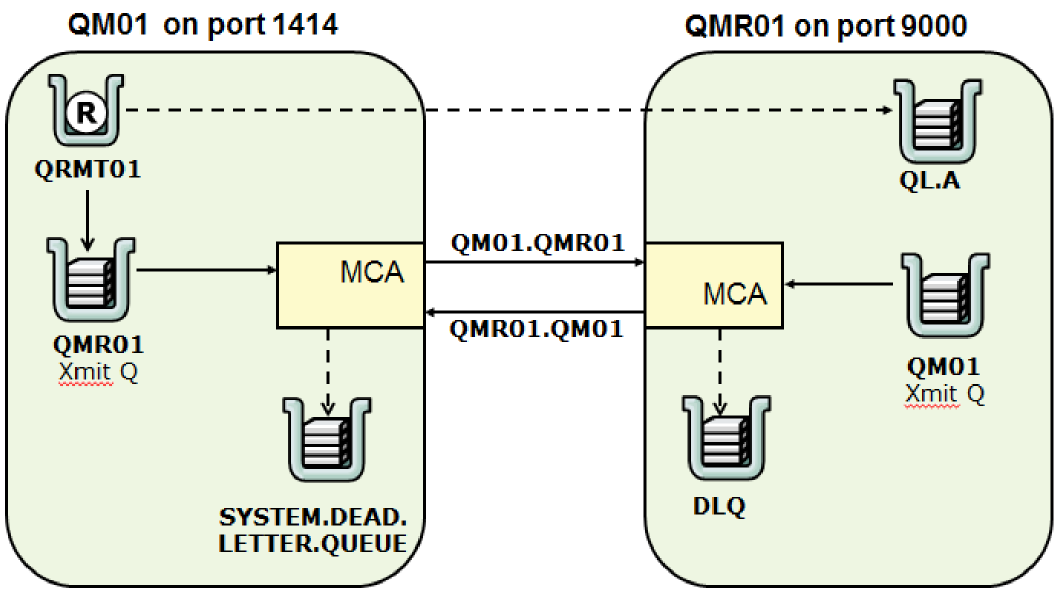
- Configure a distributed network of two or more interconnected queue managers
- Use IBM MQ commands to create the channels and supporting objects to implement distributed queuing
- Use the IBM MQ sample programs to test the connection between the queue managers

Introduction

In this exercise, you must configure two queue managers. One of queue managers (QM01) is a local queue manager, and the other queue manager (QMR01) **simulates** a queue manager on a remote server.

In Part 1 of this exercise, you create a server-connection channel on QMR01 so that you can manage the queue manager as if it is a queue manager on another server. You learn more about server-connection channels later in this course.

In this exercise, you configure two message channels between the queue managers so that messages can flow in each direction. The local queue manager is QM01 that you created in Exercise 1. You create queue manager QMR01 on the simulated remote server in this exercise. The following figure summarizes the configuration requirements.



In IBM MQ, channel authentication and connection authentication are enabled by default for all new queue managers. By default, channel authentication and connection authentication prevent remote connections to the queue manager from MQ Explorer. In one step of this exercise, you disable channel authentication and connection authentication on the queue manager to allow connections to remote queue managers from MQ Explorer.

The exercise uses TCP/IP as the communications protocol.

Requirements

- IBM MQ and IBM MQ Explorer
- The queue manager QM01 and queues that were created in Exercise 1
- Familiarity with the sample programs that were described in Exercise 3
- The IBM MQ sample programs **amqsput**, **amqsget**, **amqsreq**, and **amqsech**
- A text editor

Exercise instructions

Part 1: Create a remote queue manager

In this part of the exercise, you create queue manager QMR01.

You gain experience with the steps that are required to connect to a queue manager on a remote server from MQ Explorer. You connect to this queue manager from MQ Explorer as if it is on a remote server.

__ 1. Start the queue manager QM01 that you created in Exercise 1 unless it is already running.

__ 2. Using MQ Explorer, create a queue manager that is named **`QMR01`** that uses a queue that is named **`DLQ`** for a dead-letter queue and a listener that uses port number 9000.

__ 3. Using MQ Explorer, create the local queue DLQ for the queue manager dead-letter queue (defined in the previous step).

__ 4. In IBM MQ, channel authentication and connection authentication are enabled by default for all new queue managers. By default, channel authentication and connection authentication prevent remote connections to the queue manager from MQ Explorer.

Use MQSC to disable channel authentication and connection authentication on QMR01 to allow a remote connection from MQ Explorer.

__ a. In a command window, type: **`runmqsc QMR01`**

__ b. Display the queue manager properties. Type: **`DIS QMGR`**

__ c. Verify that the CHLAUTH attribute is set to **`ENABLED`** and that the CONNAUTH attribute is set to **`SYSTEM.DEFAULT.AUTHINFO.IDPWOS`**. These settings are the default values.

__ d. Disable channel authentication. Type:

```
ALTER QMGR CHLAUTH(DISABLED)
```

__ e. Verify that the CHLAUTH attribute is now set to **`DISABLED`**.

__ f. For this exercise, disable client connection authentication. Type:

```
ALTER QMGR CONNAUTH(' ')
```

Note

A space character is between the single quotation mark characters in the **`CONNAUTH`** attribute of the **`ALTER QMGR`** command.

__ g. Refresh the security cache for the queue manager. Type:

```
REFRESH SECURITY
```

__ h. End the MQSC session. Type: **`END`**

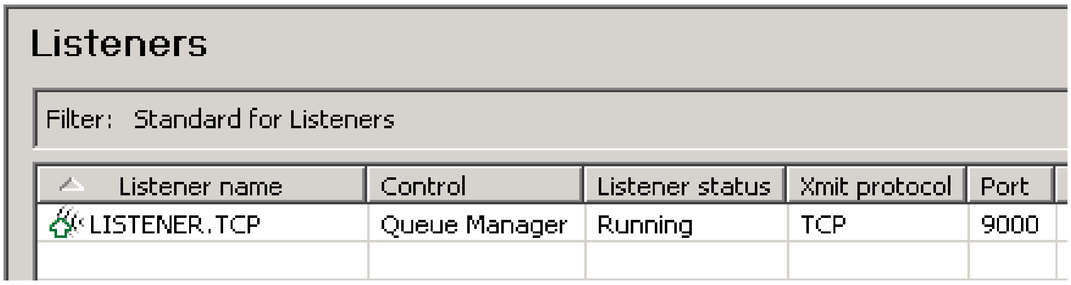
Important

The steps to disable channel and connection authentication are not standard practice. Authentication is disabled in this exercise because security configuration is covered in a later unit.

__ 5. Verify that the listener is running on QMR01.

__ a. In the **MQ Explorer - Navigator** view, click the **Listeners** folder under QMR01.

__ b. Verify that the listener that is named **LISTENER.TCP** is in the list, that the status is **Running**, and that the listener is running on port 9000.



Listeners

Filter: Standard for Listeners

Listener name	Control	Listener status	Xmit protocol	Port
LISTENER.TCP	Queue Manager	Running	TCP	9000

__ 6. In this step, you configure the queue manager QMR01 so that you can connect to it from MQ Explorer as if the queue manager is on a remote server.

In MQ Explorer, create a server connection channel (SVRCONN) on the queue manager QMR01.

__ a. In the **MQ Explorer - Navigator** view, right-click the queue manager QMR01 and then click **Remote Administration**.

__ b. In the **Remote Administration** window, click **Create** to create the SYSTEM.ADMIN.SVRCONN channel.

Click **OK** on the confirmation window and then click **Close**.

Information

You can display the SYSTEM channels in MQ Explorer by clicking **Channels** in the **MQ Explorer - Navigator** view and then clicking the **Show System Objects** icon in the **Channels** view.

__ 7. Connect to the queue manager QMR01 from MQ Explorer as if it is running on a remote server.

__ a. In the **MQ Explorer - Navigator** view, right-click **Queue Managers** and then click **Add Remote Queue Manager**.

__ b. In the **Add Queue Manager** window, type: `QMR01`

__ c. Select **Connect directly** and then click **Next**.

__ d. For the **Host name or IP address**, type: `localhost`

__ e. For the **Port number**, type: `9000`

__ f. Verify that the server-connection channel is SYSTEM.ADMIN.SVRCONN.

__ g. Click **Finish.**

The queue manager should now be visible in the **MQ Explorer - Navigator** view as **QMR01 on 'localhost(9000)'**.

Part 2: Define the channels and listener on the local queue manager

In this part of the exercise, you define a sender channel and a receiver channel on queue manager, QM01.

__ 1. Using a text editor, create an MQSC command file that is named **`ConnectLocal.mqsc`** to define the MQ objects that are required for a connection between QM01 and QMR01.

__ a. Define a message sender channel.

- Channel name = **`QM01.QMR01`** (where `QM01` is the local queue manager name and `QMR01` is the remote queue manager)
- Protocol = **`TCP/IP`**
- Network address of QMR01 = **`localhost(9000)`** (where `localhost` is the host name and 9000 is the listener port)
- Transmission queue name = **`QMR01`** (the same as the name of the remote queue manager)

Type:

```
DEF CHL(QM01.QMR01) CHLTYPE(SDR) REPLACE +
TRPTYPE(TCP) CONNAME('localhost(9000)') +
XMITQ(QMR01)
```

__ b. Define a receiver channel. The attributes must match the sender channel of the remote queue manager QMR01. Type:

```
DEF CHL(QMR01.QM01) CHLTYPE(RCVR) REPLACE +
TRPTYPE(TCP)
```

__ c. Create a transmission queue with the same name as the remote queue manager QMR01. Type:

```
DEF QL(QMR01) REPLACE USAGE(XMITQ)
```

__ 2. Use **`runmqsc`** to run the script file **`ConnectLocal.mqsc`** against the local queue manager QM01. Redirect the output to a text file that is named **`LocalReport.txt`**.

__ a. Type: **`runmqsc QM01 < ConnectLocal.mqsc > LocalReport.txt`**

Note

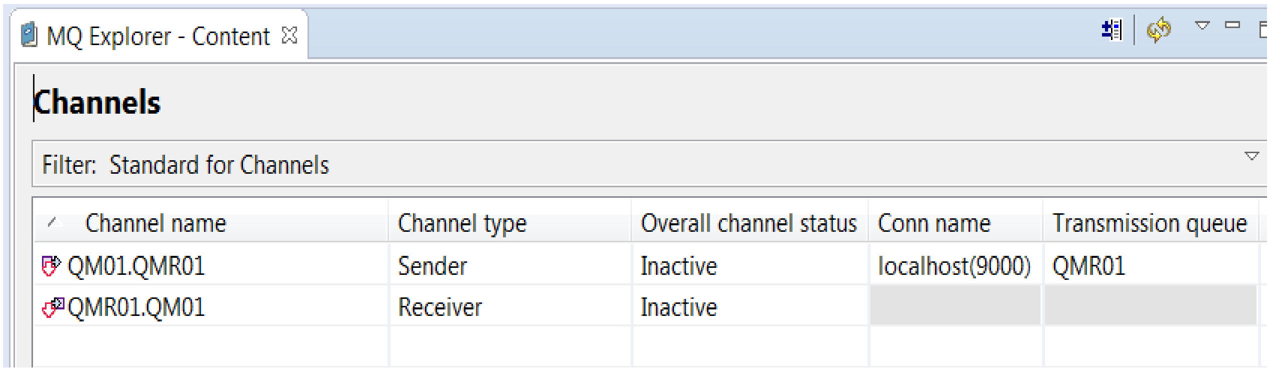
If you did not save the **`ConnectLocal.mqsc`** file in the current directory, you must either change directories or specify the path name. For example:

```
runmqsc QM01 < /labfiles/Lab04/ConnectLocal.mqsc > /labfiles/Lab04/LocalReport.txt
```

__ 3. Examine the **`LocalReport.txt`** file and verify that all commands ran successfully.

__ 4. Use MQ Explorer to verify that QM01 now has two channels:

- A sender channel that is named **`QM01.QMR01`**
- A receiver channel that is named **`QMR01.QM01`**



MQ Explorer - Content

Channels

Filter: Standard for Channels

Channel name	Channel type	Overall channel status	Conn name	Transmission queue
QM01.QMR01	Sender	Inactive	localhost(9000)	QMR01
QMR01.QM01	Receiver	Inactive		

__ 5. Verify that QM01 now has a transmission queue that is named `QMR01`.

__ 6. When you created QM01 in Exercise 1, you did not create or start a listener. In this step, use the `DEFINE LISTENER` command to create a listener on port 1414 for the queue manager QM01.

Use the `CONTROL(QMGR)` option in the `DEFINE LISTENER` command to associate the listener service with the queue manager. The listener service automatically starts when the queue manager starts and stops when the queue manager stops.

In MQSC for QM01, type:

```
DEF LISTENER(LISTENER.TCP) TRPTYPE(TCP) PORT(1414) CONTROL(QMGR)
```

__ 7. Use the `START LISTENER` command to start the listener on QM01.

In MQSC for QM01, type:

```
START LISTENER(LISTENER.TCP)
```

Alternatively, you can start the MQ listener by using the **`runmqlsr`** command. For example:

```
runmqlsr -t TCP -p 1414 -m QM01
```

If you use the **`runmqlsr`** command, you must use the **`endmqlsr`** command to manually stop the service.

Linux

If you want to start the MQ listener in the background on Linux, type:

```
nohup runmqlsr -m QM01 -t tcp -p 1414 2>&1 &
```

You can verify that the listener is running by typing the following command in a terminal window:

```
ps -ef | grep runmqlsr
```

__ 8. To eliminate any security errors and security configuration in this exercise, disable the use of channel authentication and connection authentication on QM01.

In MQSC for QM01, type:

```
ALTER QMGR CHLAUTH(DISABLED)
ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) +
AUTHTYPE(IDPWOS) CHCKCLNT(NONE)
REFRESH SECURITY
```

Important

The steps to disable channel and connection authentication are not standard practice. Authentication is disabled in this exercise because security configuration is covered in a later unit.

Part 3: Define the channels on the remote queue manager

In this part of the exercise, you define the sender channel, receiver channel, and transmission queue for QMR01.

__ 1. Using a text editor, create an MQSC command file that is named **`ConnectRemote.mqsc`** to define the MQ objects that are required for a connection between your local queue manager and the remote queue manager.

__ a. Define a sender channel that matches the receiver channel on QM01.

- Channel name = **`QMR01.QM01`**
- Protocol = **`TCP/IP`**
- Network address of QM01 = **`localhost(1414)`**
- Transmission queue name = **`QM01`**

Type:

```
DEF CHL(QMR01.QM01) CHLTYPE(SDR) REPLACE +
TRPTYPE(TCP) CONNAME('localhost(1414)') +
XMITQ(QM01)
```

__ b. Define a receiver channel. The attributes must match the sender channel QM01. Type:

```
DEF CHL(QM01.QMR01) CHLTYPE(RCVR) REPLACE +
TRPTYPE(TCP)
```

__ c. Create a transmission queue for QM01. Type:

```
DEF QL(QM01) REPLACE USAGE(XMITQ)
```

__ 2. Use **`runmqsc`** to run the script file **`ConnectRemote.mqsc`** against the remote queue manager QMR01. Redirect the output to a text file that is named **`RemoteReport.txt`**.

Type: **`runmqsc QMR01 < ConnectRemote.mqsc > RemoteReport.txt`**

__ 3. Use the report and MQ Explorer to verify that QMR01 now has two channels:

- A receiver channel that is named **`QM01.QMR01`**
- A sender channel that is named **`QMR01.QM01`**

Also, verify that QMR01 now has a transmission queue that is named **`QM01`**.

Part 4: Test and start the connection

__ 1. To test the channel definitions, open another command window, and use the **`runmqsc`** command to ping the message channel from the QM01 (the sender). Type:

```
runmqsc QM01
PING CHL(QM01.QMR01)
```

Check for successful completion.

__ 2. In MQSC for QM01, use the **`START CHANNEL`** command to start the sender channel and verify that it is working.

Type: **`START CHL(QM01.QMR01)`**

__ 3. Using MQSC for QM01, verify that the channel is running.

Type: **`DIS CHSTATUS(*)`**

You should see that the channel QM01.QMR01 is running, the remote queue manager name is QMR01, and the transmission queue is QMR01.

Part 5: Test queues in a distributed environment

In this part of the exercise, you create the queues for testing queue managers in a distributed environment and then use the MQ sample programs to put and get messages.

__ 1. On QMR01, define a local queue that is named **QL.A**.

In MQSC for QMR01, type: **`DEF QL(QL.A) REPLACE`**

__ 2. In MQSC, define the application queues on QM01.

__ a. Delete and redefine the local queue **QL.A**. Type:

```
DEF QL(QL.A) REPLACE
```

__ b. Verify that you defined a transmission queue that is named QMR01. Type:

```
DIS QL(QMR01)
```

Verify that the USAGE attribute is set to **`XMITQ`**.

__ c. Create a remote queue definition that is named **`QRMT01`** that points to the local queue QL.A on QMR01 and uses QMR01 for the transmission queue.

Type the following command:

```
DEF QR(QRMT01) REPLACE +
RNAME(QL.A) RQMNAME(QMR01) +
XMITQ(QMR01)
```

__ d. Verify the remote queue definition. Type:

```
DIS QR(QRMT01)
```

__ 3. Use the sample program **`amqsput`** to send messages from queue manager QM01 to the queue QL.A on queue manager QMR01.

In a command window, type:

```
amqsput QRMT01 QM01
```

__ 4. Use MQ Explorer to verify that the messages you sent in Step 3 are on queue QL.A on QMR01.

You can also use MQSC to check the CURDEPTH attribute of the target queue QL.A on QMR01.

__ 5. If the messages do not arrive on the target, investigate the possible causes. Check the following items:

__ a. Is the transmission queue on QM01 empty?

__ b. Is the sender channel on QM01 running?

__ c. Is the dead-letter queue of the remote queue manager QMR01 empty?

__ d. Inspect the error logs in both queue managers.

Exercise cleanup

__ 1. Clear the messages on QL.A on the QMR01 queue manager.

__ 2. Leave the queue managers running. You use these queue managers in the next exercise.

End of exercise

Exercise review and wrap-up

Having completed this exercise, you should be able to:

- Create the objects that are required for distributed queuing
- Configure and refresh the started MQ listener
- Start message channels manually
- Use sample applications to test the message flow

Exercise 5. Connecting an IBM MQ client

Estimated time

01:00

Overview

In this exercise, you configure your system to act as a client that is connected to an IBM MQ server. You use various methods to gain experience with the client connectivity methods that are available in IBM MQ.

Objectives

After completing this exercise, you should be able to:

- Create a server connection channel to support client connections
- Use a URL to specify the location of the client connection definition table
- Use the MQSERVER environment variable to specify a client connection channel
- Use the client configuration file to specify a client connection channel

Introduction

As you learned in lecture, you can use different methods to establish a connection between an MQ client and an MQ server. In this exercise, you use the different methods to give you experience with each. In an actual implementation, you would use one of these methods.

In this exercise, you use the client sample programs to put, browse, and get messages by using a client connection. A brief description of each of the programs you might use is provided:

- **`amqsputc`** puts a message to a queue. It is started the same way as **`amqsput`** and uses the same parameter structure, but it connects as an MQ client instead of a using a direct connection to the MQ server.
- **`amqsbcgc`** browses the messages on a queue. It is started the same way as **`amqsbcg`** and uses the same parameter structure, but it connects as an MQ client.
- **`amqsgetc`** gets the messages from a queue. It is started the same way as **`amqsget`** and uses the same parameter structure, but it connects as an MQ client.

To simplify running this exercise, you should use two console windows. Use one of the console windows for running the sample programs. Use the other console window for entering queue manager control and configuration commands.

Requirements

- IBM MQ and IBM MQ Explorer.
- IBM MQ sample programs
- The queue manager QM01 that was created in Exercise 1 and modified in Exercise 4 to disable authentication.
- On Windows, the `mqclient.ini` file in the **`C:\labfiles\Lab05`** directory.

Exercise instructions

In this exercise, the queue manager QM01 is configured to act as a server for the MQ clients. You use the MQ client sample programs to connect the MQ client to the queue manager QM01.

Important

For this exercise, channel authentication must be disabled on QM01 as instructed in Exercise 4.

Server queue manager setup

__ 1. Reuse the queue manager that is defined in Exercises 1 and 4 (QM01) as a server queue manager.

On the server queue manager QM01, define a server connection (SVRCONN) channel so that MQ clients can connect to the queue manager. When you define the server-connection channel, ensure that the channel definition meets the following requirements:

- Use **QM01_CLNT** as the channel name. The protocol is TCP.
- To ensure that the user has the object authority on the target server, add the MCA user identifier option MCAUSER.
- If the queue manager is running on Windows, set MCAUSER to: `MUSR_MQADMIN`
- If the queue manager is on Linux, set MCAUSER to: **`mqm`**

Windows

To define the SVRCONN on QM01, type:

```
DEFINE CHL(QM01_CLNT) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
MCAUSER(MUSR_MQADMIN)
```

Linux

To define the SVRCONN on QM01, type:

```
DEFINE CHL(QM01_CLNT) CHLTYPE(SVRCONN) TRPTYPE(TCP)+
MCAUSER('mqm')
```

Important

In this lab, you are using MUSR_MQADMIN or “mqm” for the MCAUSER user name. In a real implementation, you would create an administrative user ID for MQ and use that ID for the MCAUSER. Never use MUSR_MQADMIN or “mqm” in a real implementation. You learn more about security later in this course.

__ 2. Verify the channel that you defined in Step 1. Type:

```
DIS CHL(QM01_CLNT)
```

You can also verify the channel by using MQ Explorer.

__ 3. On the server queue manager QM01, define another server connection (SVRCONN) channel that you use later in this exercise to connect by using a URL.

Use **FILEURL** as the channel name. The protocol is TCP.

To ensure that the user has the object authority on the target server, add the MCA user identifier option MCAUSER.

W **Windows**

To define the SVRCONN and set the MCAUSER to `MUSR_MQADMIN` on QM01, type:

```
DEFINE CHL(FILEURL) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
MCAUSER(MUSR_MQADMIN)
```

L **Linux**

To define the SVRCONN and set the MCAUSER to **mqm** on QM01, type:

```
DEFINE CHL(FILEURL) CHLTYPE(SVRCONN) TRPTYPE(TCP)+
MCAUSER('mqm')
```

__ 4. Verify the channel that you defined in Step 3. Type:

```
DIS CHL(FILEURL)
```

You can also verify the channel by using MQ Explorer.

__ 5. In Exercise 4, you defined a TCP listener on port 1414 for QM01. Verify that the listener is running. In MQSC for QM01, type:

```
DIS LSSTATUS(LISTENER.TCP)
```

The listener STATUS should be RUNNING and the PORT should be 1414.

If the listener is not running, start the MQ listener on QM01 on port 1414 by typing:

```
START LISTENER(LISTENER.TCP)
```

Method 1: Client connection by using a file URL

In this part of the exercise, you connect to the queue manager QM01 by using a client connection channel that is defined on the client. The client locates the CCDT by using the file URL that is specified in the MQCCDTURL environment variable.

__ 1. Create a directory to store the client CCDT table.

W **Windows**

Create a subdirectory under the **`C:\labfiles`** directory that is named: **`ccdturl`**

L **Linux**

Create a subdirectory under **`/home/localuser/labfiles`** that is named: **`ccdturl`**

__ 2. On the client CCDT table, define a client connection with the same name as the server connection on QM01 (FILEURL).

__ a. Start a client MQSC session. Type:

```
runmqsc -n
```

The command should return:

```
Starting local MQSC for 'AMQCLCHL.TAB'.
```

__ b. Verify that the CCDT table on the client is empty. Type:

```
DIS CHL(*)
```

The command should return:

```
File 'AMQCLCHL.TAB' not found.
```

__ c. Create the client connection channel that is named FILEURL that connects to QM01. Type:

```
DEF CHL(FILEURL) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
CONNAME('localhost(1414)')
```

__ d. Verify that the channel is now defined in the CCDT table. Type:

```
DIS CHL(*)
```

The command should return:

```
CHANNEL(FILEURL) CHLTYPE(CLNTCONN)
```

__ e. Exit client MQSC mode.

__ 3. Copy the CCDT table file **`AMQCLCHL.TAB`** from the default location to the **`ccdturl`** subdirectory that you created in Step 1.

Windows

On your image, the default location for the `AMQCLCHL.TAB` file is the MQ data directory: `C:\ProgramData\IBM\MQ`

Linux

On your image, the default location for the `AMQCLCHL.TAB` file is MQ data directory: `/var/mqm`

__ 4. Set the MQCCDTURL environment variable to point to the CCDT that contains the definition of channel FILEUR by using a file URL.

Type the following commands in the same command window that you use to run the MQ sample client programs.

Windows

Type: `SET MQCCDTURL=file://C:/labfiles/ccdturl/AMQCLCHL.TAB`

Confirm that the `MQCCDTURL` variable was set correctly. Type: `echo %MQCCDTURL%`

Linux

Type: `export MQCCDTURL=file:///home/localuser/labfiles/ccdturl/AMQCLCHL.TAB`

Confirm that the `MQCCDTURL` variable was set correctly. Type: `echo $MQCCDTURL`

__ 5. Use the **amqsputc** sample program to put messages on the queue QL.A on QM01. Type:

```
amqsputc QL.A
```

__ 6. While the **amqsputc** client program is running in the command window, verify that server connection channel FILEURL is running on QM01 in another window. In MQSC for QM01, type:

```
DIS CHSTATUS(FILEURL)
```

You can also check the channel status in MQ Explorer by right-clicking the channel in the **Channels** view and then clicking **Status**.

__ 7. In the test window where **amqsputc** is running, put some messages to QL.A.

__ 8. Using MQ Explorer, verify that your messages are on the QL.A on QM01.

__ 9. Use the **amqsbcgc** client program to browse the messages on the server queue. Type:

```
amqsbcgc QL.A
```

The value of **ReplyToQmgr** in the MQMD shows you the queue manager to which the client connected.

__ 10. Use the **amqsgetc** program to get the messages from QL.A and empty the queue. Type:

```
amqgetc QL.A
```

__ 11. Clear the **MQCCDTURL** environment variable.

Windows

From the command prompt, type: **`SET MQCCDTURL=`**

Linux

In the terminal window, type: **`unset MQCCDTURL`**

Method 2: Client connection by using the MQSERVER environment variable

In this part of the exercise, use the MQSERVER environment variable to specify a client connection channel.

__ 1. Use the MQSERVER environment variable to provide a client-connection channel definition to connect to the server queue manager (QM01) by using the QM01_CLNT channel.

Enter the following commands according to your system type.

Windows

```
SET MQSERVER=QM01_CLNT/TCP/localhost(1414)
```

Confirm that the `MQSERVER` variable was set correctly. Type: **echo `%MQSERVER%`**

Linux

```
export MQSERVER=QM01_CLNT/TCP/'localhost(1414)'
```

Confirm that the `MQSERVER` variable was set correctly. Type: **echo `$MQSERVER`**

__ 2. In the same window that you set the **`MQSERVER`** environment variable, test the client connection. Use the **amqsputc** sample program to put messages on the queue QL.A on the QM01 queue manager. Type:

```
amqsputc QL.A
```

__ 3. While the **amqsputc** client program is running in the command window, verify that server connection channel QM01_CLNT is running on QM01 in another window.

In MQSC for QM01, type:

```
DIS CHSTATUS(QM01_CLNT)
```

You can also check the channel status in MQ Explorer by right-clicking the channel in the **Channels** view and then clicking **Status**.

__ 4. In the test window where **`amqsputc`** is running, put some messages to QL.A.

__ 5. Using MQ Explorer, verify that your messages are on the QL.A on QM01.

__ 6. Use the **`amqsbcgc`** client program to browse the messages on the server queue. Type:

```
amqsbcgc QL.A
```

The value of **ReplyToQmgr** in the MQMD shows you the queue manager to which the client connected.

__ 7. Use the **`amqsgetc`** program to get the messages from QL.A and empty the queue. Type:

```
amqsgetc QL.A
```

__ 8. Clear the MQSERVER environment variable.

L **Linux**

In a terminal window, type: **`unset MQSERVER`**

W **Windows**

From a command prompt, type: **`SET MQSERVER=`**

Method 3: Client connection by using the client configuration file

In this part of the exercise, you use the client configuration file to specify information about the client channels. The template for the client configuration file that is named **`mqclient.ini`** is provided for you in the lab files on the image.

W **Windows**

The client configuration file **`mqclient.ini`** is in the **`C:\labfiles\Lab05`** directory.

L **Linux**

The client configuration file **`mqclient.ini`** is in the **`/var/mqm`** directory.

__ 1. Open the client configuration file **`mqclient.ini`** in a text editor.

__ a. Edit the **`ServerConnectionParms`** attribute to specify the connection to the QM01_CLNT channel on QM01. Type:

```
QM01_CLNT/TCP/localhost(1414)
```

Example:

```
CHANNELS:
    DefRecon=YES
    ServerConnectionParms=QM01_CLNT/TCP/localhost(1414)
```

__ b. Save and close the file.

__ 2. Use the MQCLNTCF environment variable to identify the location of the client configuration file.

Enter the following commands according to your system type.

Windows

```
SET MQCLNTCF="C:\labfiles\Lab05\mqclient.ini"
```

Linux

```
export MQCLNTCF=/var/mqm/mqclient.ini
```

__ 3. Test the client connection.Use **amqsputc** to put messages on the queue QL.A on the QM01 queue manager. Type:

```
amqsputc QL.A
```

__ 4. While the **amqsputc** client program is running in the command window, verify that server connection channel QM01_CLNT is running on QM01 in another window. In MQSC for QM01, type:

```
DIS CHSTATUS(QM01_CLNT)
```

You can also check the channel status in MQ Explorer by right-clicking the channel in the **Channels** view and then clicking **Status**.

__ 5. In the test window where **amqsputc** is running, put some messages to QL.A.

__ 6. Using MQ Explorer, verify that your messages are on the QL.A on QM01.

__ 7. Use the **amqsbcgc** client program to browse the messages on the server queue. Type:

```
amqsbcgc QL.A
```

The value of **ReplyToQmgr** in the MQMD shows you the queue manager to which the client connected.

__ 8. Use the **amqsgetc** program to get the messages from QL.A and empty the queue. Type:

```
amqsgetc QL.A
```

__ 9. Clear the MQCLNTCF environment variable.

Windows

From a command prompt, type: `SET MQCLNTCF=`

Linux

In a terminal window, type: `unset MQCLNTCF`

End of exercise

Exercise review and wrap-up

In the absence of any security exits, the IBM MQ client uses a field in the channel descriptor to pass the client logged-on user ID to the server where it is used for access control.

The problem with this approach is that *spoofing*, or pretending to be someone you are not, is easy. If you have administrative control of the client system, you can define a user ID, such as `mqm` or `MUSR_MQADMIN`. You can then use that ID to connect to a remote queue manager with full access to that queue manager. This approach is not satisfactory.

Having completed this exercise, you should be able to:

- Create a server connection channel to support client connections
- Use a URL to specify the location of the client connection definition table
- Use the MQSERVER environment variable to specify a client connection channel
- Use the client configuration file to specify a client connection channel

Exercise 6. Implementing a trigger monitor

Estimated time

01:00

Overview

In this exercise, you modify queue parameters to implement triggering. You define a process object that identifies the IBM MQ **echo** sample program and triggers the program when a message arrives on a queue. You use the **request** sample program to send messages to the queue and retrieve the responses.

Objectives

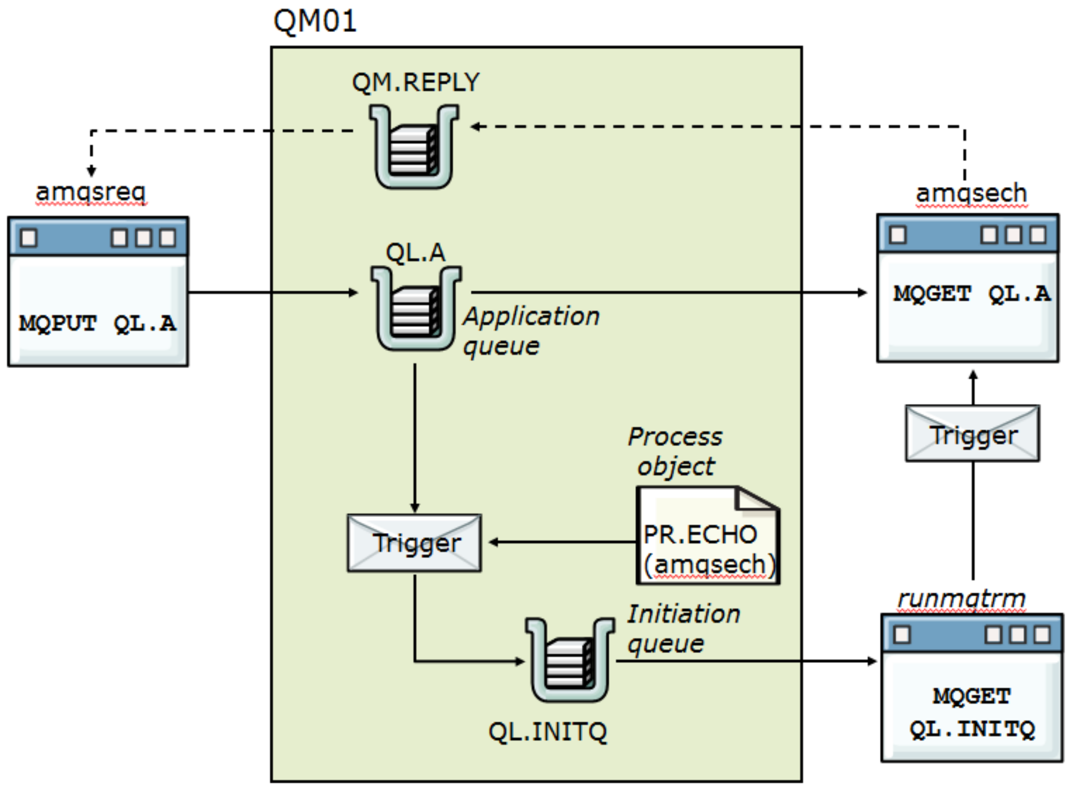
After completing this exercise, you should be able to:

- Apply triggering parameters to queues
- Start a trigger monitor
- Test triggering by using IBM MQ sample programs

Introduction

In this exercise, you modify the parameters of a queue to implement triggering. You then define a process object that identifies an IBM MQ sample program (**amqsech**) that is triggered when a message arrives on that queue. You use the **amqsreq** sample program to send messages to the queue and retrieve the responses. The trigger monitor starts the other sample programs and builds the reply messages.

The following figure shows the IBM MQ objects that you create and how the sample programs are used in this exercise.



- The Request sample program **amqsreq** puts the message on the application queue QL.A and gets a reply message from the QM.REPLY queue.

  The command to start the program is: `amqsreq` *`Qname QmgrName replyto_qname`*

  Example: **`amqsreq QL.A QM01 QM.REPLY`**.

  The program reads lines of text from the standard input device (the terminal windows), converts them to request messages, and then puts the messages on the named queue.

  When a null line is entered, the program gets the messages from the reply-to queue and displays those messages on the standard output device.

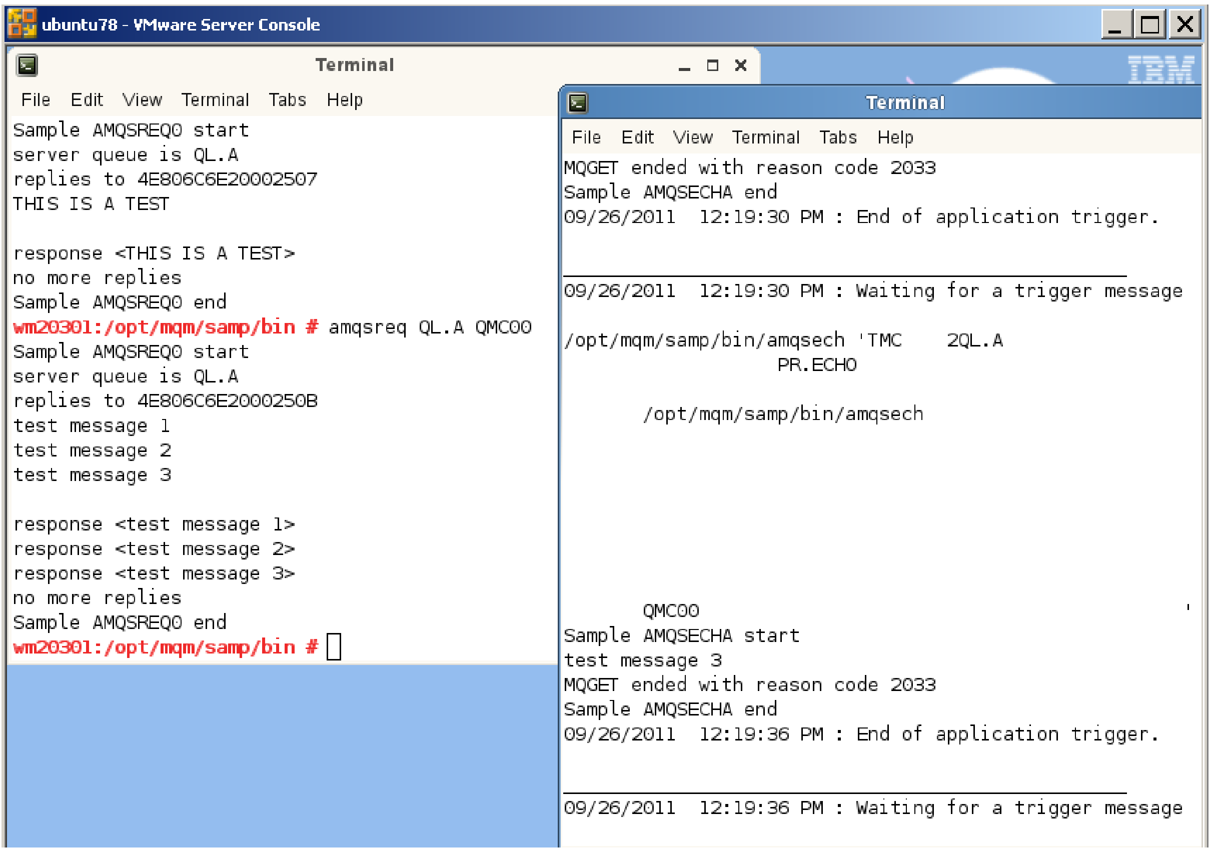
  The **amqsreq** program stops when the queue is empty.

- A trigger monitor starts the Echo program **amqsech**. The **amqsech** program connects to the queue manager that is named in the structure that is passed to it by the trigger monitor and then opens the queue that is also named in the structure.

  The **amqsech** program gets a message from the application queue, QL.A, and creates a message with the same application data as the original message. It puts the new message on the reply-to queue named in the message descriptor of the original message, QM.REPLY. The program then gets each of the remaining messages on the request queue in turn and generates a reply in the same way. When the request queue is empty, the program closes the queue and disconnects from the queue manager.

The following screen capture of a Linux image provides an example of how to arrange your terminal windows so that you see the **amqsreq** program in one window and the trigger monitor in another window.

Upon successful completion of Part 2 of this exercise, you should see that when you run the amqsreq program, the trigger monitor window indicates that the amqsech sample program starts.


ubuntu78 - VMware Server Console
Terminal
File Edit View Terminal Tabs Help
Terminal
File Edit View Terminal Tabs Help

```
Sample AMQSREQ0 start
server queue is QL.A
replies to 4E806C6E20002507
THIS IS A TEST

response <THIS IS A TEST>
no more replies
Sample AMQSREQ0 end
wm20301:/opt/mqm/samp/bin # amqsreq QL.A QMC00
Sample AMQSREQ0 start
server queue is QL.A
replies to 4E806C6E2000250B
test message 1
test message 2
test message 3

response <test message 1>
response <test message 2>
response <test message 3>
no more replies
Sample AMQSREQ0 end
wm20301:/opt/mqm/samp/bin #
```

```
MQGET ended with reason code 2033
Sample AMQSECHA end
09/26/2011  12:19:30 PM : End of application trigger.

___________________________________________________________________
09/26/2011  12:19:30 PM : Waiting for a trigger message

/opt/mqm/samp/bin/amqsech 'TMC    2QL.A
                 PR.ECHO

      /opt/mqm/samp/bin/amqsech






      QMC00
Sample AMQSECHA start
test message 3
MQGET ended with reason code 2033
Sample AMQSECHA end
09/26/2011  12:19:36 PM : End of application trigger.

___________________________________________________________________
09/26/2011  12:19:36 PM : Waiting for a trigger message
```

In Part 3 of this exercise, a trigger monitor starts the **amqsinq** program. The program connects to the queue manager, opens the request queue, gets a message from the queue, and interprets the application data as the name of a queue. It then constructs a reply message with the attribute values, and puts the message on the reply-to queue. The **amqsinq** program continues to get each of the remaining messages on the request queue and generates a reply. When the request queue is empty, the program closes the queue and disconnects from the queue manager.

It is not necessary for you to stop the trigger monitor at the end of this lab; changes can be made while the trigger monitor is running. The trigger monitor stops with the queue manager. You can force the trigger monitor to stop by closing the window in which the trigger monitor was running or use Ctrl+C to end the process.

Requirements

- IBM MQ and IBM MQ Explorer
- The sample programs **amqsreq**, **amqsech**, and **amqsinq**
- A text editor
- The queue manager QM01 and queues that were created in Exercise 1

Exercise instructions

In this exercise, you modify the parameters of a queue to implement triggering. You then define a process object that identifies an IBM MQ sample program (**amqsech**) that is triggered when a message arrives on that queue. You use the **amqsreq** sample program to send messages to the queue and retrieve the responses. The trigger monitor starts the other sample programs and builds the reply messages.

Part 1: Configure the queue manager for triggering

__ 1. Prepare an MQSC command file that is named **`trig.mqsc`** to create all the following objects on the default queue manager QM01.

Use the REPLACE parameter on all relevant MQSC commands so that the command file can be run again if necessary.

__ a. Create an initiation queue that is named QL.INITQ.

Type: **`DEFINE QL(QL.INITQ) REPLACE`**

__ b. Alter the local queue that is named QL.A to enable triggering. The arrival of the first message on QL.A triggers the appropriate application. Make sure that the PUT and GET operations are enabled, and clear any messages that might be on the queue from a previous lab.

Type:

```
DEFINE QL(QL.A) REPLACE +
TRIGGER TRIGTYPE(FIRST) +
PROCESS(PR.ECHO) INITQ(QL.INITQ)
```

__ c. Define a process to identify the application to start. Define the process so that **amqsech** is started synchronously with the trigger monitor. Name the process: `PR.ECHO`

Use one of the following commands that are based on your system type.

Windows

```
DEFINE PROCESS(PR.ECHO) REPLACE +
APPLICID('amqsech')
```

Linux

```
DEFINE PROCESS(PR.ECHO) REPLACE +
APPLICID('/opt/mqm/samp/bin/amqsech')
```

__ d. Define a model queue that is named QM.REPLY. The sample program **amqsreq** opens this queue to create a temporary dynamic reply-to queue.

Type: **`DEFINE QM(QM.REPLY) REPLACE`**

__ 2. Use **runmqsc** to process the command file. Open the report in a text editor and verify that all the objects are successfully created. If unsuccessful, edit and rerun the command file until all the objects are created.

Type: **`runmqsc QM01 < trig.mqsc > trigreport.txt`**

Information

You can verify the process definition in MQ Explorer by clicking the **Process Definitions** folder under the queue manager in the **MQ Explorer - Navigator** view.

You can modify the process definition in MQ Explorer by double-clicking the process name in the **Process Definitions** view.

__ 3. Use the **`runmqtrm`** control command to start the trigger monitor. Specify your queue manager and the name of the initiation queue.

Type: **`runmqtrm -q QL.INITQ -m QM01`**

You should see that the trigger monitor program is running and waiting for a message.

Do not close this window. If you close this window, the trigger monitor stops.

Part 2: Test triggering

__ 1. Open a new command window and run **amqsreq** to put some request messages on the local queue that is enabled for triggering.

Type: **`amqsreq QL.A QM01 QM.REPLY`**

__ 2. Type a message, and press Enter.

For example, type: `Hello this is a message from me`

__ 3. Press Enter again to end the sample program.

__ 4. Verify that the **amqsreq** program starts and that you received a response message in the command window.

The response message is similar to the following message:

```
response <Hello this is a message from me>
```

The program continues to run a few seconds and the following message is displayed:

```
no more replies
Sample QMQSREQ0 end
```

__ 5. In the trigger monitor window, verify that the **amqsech** program started and received the message.

If triggering failed, investigate the possible causes. Was one of the conditions for a trigger event not satisfied?

Part 3: Test triggering with amqsinq

__ 1. In MQSC, create a second process object on QM01 so that the trigger monitor starts **amqsinq** instead of **amqsech**.

W Windows

```
DEFINE PROCESS(PR.INQ) REPLACE +
APPLICID('amqsinq')
```

L Linux

```
DEFINE PROCESS(PR.INQ) REPLACE +
APPLICID('/opt/mqm/samp/bin/amqsinq')
```

__ 2. Alter the queue QL.A to reference the new process.

In MQSC, type: **`ALTER QL(QL.A) PROCESS(PR.INQ)`**

__ 3. Run **amqsreq** again. The **amqsinq** program now expects the application data in each message to contain the name of a queue. For example, you can use the names of the following queues:

```
QL.A
QM.REPLY
SYSTEM.ADMIN.COMMAND.QUEUE
```

Type: **`amqsreq QL.A QM01 QM.REPLY`**

Type a valid queue name.

Press Enter again to end the sample program.

__ 4. Verity that the trigger monitor starts **amqsinq** in the trigger monitor window.

__ 5. In the window where you entered the request, verify that **amqsreq** receives the request. The replies consist of an *Inquire* on attributes of the named queues.

Your results should appear similar to the following example:

```
C:\> amqsreq QL.A QM01 QM.REPLY
Sample AMQSREQ0 start
server queue is QL.A
replies to AMQ.537CB40C021B0220
QL.A

response <QL.A has 0 messages, used by 1 jobs>
no more replies
Sample AMQSREQ0 end
```

The trigger monitor window should show that **amqsinq** is the process that started.

__ 6. Alter the process object so the trigger monitor starts **amqsinq** as a separate asynchronous process.

In MQSC for QM01, type the following command to modify the process definition.

Windows

```
ALTER PROCESS(PR.INQ) APPLICID('start amqsinq')
```

Linux

Set the ENVRDATA attribute to a single space character followed by the ampersand character to run the application in the background. Type:

```
    ALTER PROCESS(PR.INQ) ENVRDATA(' &')
```

__ 7. Run **amqsreq** again following the directions in Step 3 and verify that the triggered application opens in a separate window (on Windows) or a background process (on Linux).

__ 8. Stop the trigger monitor (close the trigger monitor window).

Information

The trigger monitor stops with the queue manager. You can force the trigger monitor to stop by closing the window in which the trigger monitor was running or use Ctrl+C to end the process.

In a real implementation, it is not necessary for you to stop the trigger monitor; changes can be made to the queues and queue manager while the trigger monitor is running.

__ 9. Use MQSC, the **amqsget** sample program, or MQ Explorer to clear all the messages on QL.A if any exist.

End of exercise

Exercise review and wrap-up

Having completed this exercise, you should be able to:

- Apply triggering parameters to queues
- Start a trigger monitor
- Test triggering by using IBM MQ sample programs

Exercise 7. Running an IBM MQ trace

Estimated time

00:30

Overview

In this exercise, you start a trace on the IBM MQ **amqsput** and **amqsget** sample programs. You then use the sample programs to send and receive messages and examine the trace output.

Objectives

After completing this exercise, you should be able to:

- Start and stop an IBM MQ trace
- Analyze the output from the IBM MQ trace

Introduction

The trace function in MQ is used for problem determination or as a program development aid. Service personnel might request that a scenario is re-created and traced for solving error or unexpected results.

In this exercise, you start a trace on the sample programs. Then, you use a simple scenario of putting and getting a message from a local queue that is called QL.A on queue manager QM01 by using the sample programs. You also get the process ID (PID) of the sample programs, which you use to locate the trace files.

In this exercise, you have a choice of either using the commands to start and stop a trace, or using MQ Explorer for these tasks.

Attention

The trace started through MQ Explorer has some limitations. It is all or nothing; you have no choice about the traced components, even if you clear the option for “all”.

The commands that are used for controlling trace are:

- **`strmqtrc`** to enable tracing. The command includes optional parameters that specify the level of tracing. In this exercise, you use the **`-m`** parameter and the queue manager name to specify the name of the queue manager.
- **`endmqtrc`** to end a trace. In this exercise, you use the **`-m`** parameter and queue manager name to specify the name of the queue manager.
- **`dspmqtrc`** to format a trace file (UNIX and Linux only).

Sample trace files are provided with the lab files the **`Lab07\Solution`** subdirectory.

Requirements

- IBM MQ and IBM MQ Explorer.
- The exercise assumes that you defined the queue manager and queues in Exercise 1 and that you are familiar with the sample programs that were introduced in Exercise 3.
- The exercise requires sample programs **`amqsput`** and **`amqsget`**.

Exercise instructions

In this exercise, you start a trace on the IBM MQ **amqsput** and **amqsget** sample programs. You then use the sample programs to send and receive messages and examine the trace output.

Part 1: Start an MQ trace on the sample programs

__ 1. Start queue manager QM01 unless it is already running.

__ 2. Using a command or MQ Explorer, start an MQ trace on the MQ **`amqsput`** and **`amqsget`** sample programs.

When you use MQ Explorer to start a trace, you do not have the option of selecting specific components to trace. You can use the **`strmqtrc`** command to identify specific components.

To use MQ Explorer to start an MQ trace:

__ a. In the **MQ Explorer - Navigator** view, right-click the **IBM MQ** and then click **Trace**. The Trace dialog box is displayed.

__ b. Click **Start**.

W **Windows**

To start an MQ trace by using a command, type:

```
strmqtrc –m QM01 -p amqsput.exe,amqsget.exe
```

L **Linux**

To start an MQ trace by using a command, type:

```
strmqtrc –m QM01 -p amqsput,amqsget
```

Part 2: Use the sample programs to put and get messages from a queue

In this part of the exercise, you run the **amqsput** and **amqsget** sample programs, which causes the generation of trace files. You also get the process ID (PID) of the sample programs so that you can identify the trace files for each program.

__ 1. Start the **amqsput** sample program.

From a command prompt, type: **`amqsput QL.A QM01`**

Do not put any messages on the queue yet.

__ 2. Get the PID of the **amqsput** sample program.

W **Windows**

Get the PID of the **amqsput** by using the Windows Task Manager:

__ a. Right-click the Windows taskbar, and then select **Task Manager**.

__ b. Click the **Processes** tab.

__ c. Click **Image Name** to sort the processes alphabetically by program name.

__ d. If the **PID** column is not displayed, click **View > Select Columns** and then click the **PID (Process Identifier)** check-box.

__ e. Write the **amqsput** PID here: _______________

__ f. Leave the Task Manager running. You use it in Step 4 to get the PID of the **amqsget** sample program.

L **Linux**

Get the PID of the **amqsput** by using the **`ps -ef`** command.

__ a. In a second terminal window, enter: **`ps -ef | grep amqsput | grep QM01`**

__ b. Write the **amqsput** PID here: _______________

__ c. Leave this terminal window open. You are going to use it in Step 4 to get the PID of the **amqsget** sample program.

__ 3. Now enter some test messages and then press Enter on a blank line to exit the sample **amqsput** program.

__ 4. Use the **amqsget** sample program to get the messages from the local queue QL.A on queue manager QM01.

The **amqsget** displays messages in the queue, and then waits 15 seconds to see whether more messages arrive on the queue. In that 15-second period, identify the **amqsget** PID.

From a command prompt, type: **`amqsget QL.A QM01`**

W **Windows**

Find the PID for **`amqsget`** in the Windows Task Manager.

L **Linux**

In the second terminal window, enter: **`ps -ef | grep amqsget | grep QM01`**

__ a. Write the **`amqsget`** PID here: ________________

__ 5. Stop the MQ trace.

If you started the trace by using a command, stop the MQ trace in the command window. Type: **`endmqtrc –m QM01`**

If you started the trace in MQ Explorer, stop the MQ trace in MQ Explorer:

__ a. In the **MQ Explorer - Navigator** view, right-click the **IBM MQ**, and then click **Trace**. The Trace dialog box is displayed.

__ b. Click **Stop**.

Part 3: Locate and examine the trace files

W **Windows**

The default trace directory is **`C:\ProgramData\IBM\MQ\trace`**

A trace file is created for each process. The trace files are readable without formatting and are called **`AMQ`*`pppp`*`.0.TRC`** where ***pppp*** is the process identifier (PID) that created the file.

__ 1. Go to the directory **`C:\ProgramData\IBM\MQ\trace`**

__ 2. View the trace output files with Notepad and examine the PUT and GET operations.

L **Linux**

Trace files are created in the **`/var/mqm/trace`** directory. These files must be formatted before they can be read.

__ 1. Go to the directory. Type: **`cd /var/mqm/trace`**

__ 2. Format the trace files. Type: **`dspmqtrc *.TRC`**

The trace formatter program converts binary files that are named **`AMQ`*`ppppp`*`.TRC`**, where ***ppppp*** is the process identifier (PID) that created the file into readable files that are named **`AMQ`*`ppppp`*`.0.FMT`**.

__ 3. Referencing the PIDs that you captured for the **`amqsput`** and **`amqsget`** sample programs, find and view the formatted trace files, and examine the PUT and GET operations.

Sample excerpt of a Windows trace file for **amqsput**:

```
Process : C:\Program Files\IBM\MQ\Tools\c\Samples\Bin64\amqsput.exe (64-bit)
Arguments :
Host : IBM111-11111
Operating System : Windows 7 Professional x64 Edition, Build 7601: SP1
Product Long Name : IBM MQ for Windows (x64 platform)
Version : 9.0.0.0    Level : p900-L160512.4
O/S Registered : 1
Data Path : C:\ProgramData\IBM\MQ
Installation Path : C:\Program Files\IBM\MQ
Installation Name : Installation1 (1)
License Type : Production
UTC Date : 2016/10/25: Time : 14:44:12.330
LCL Date : 2016/10/25: Time : 10:44:12.330  Eastern Standard Time
QueueManager : QM01

Counter  TimeStamp          PID.TID    Ident        Data
============================================================

00000001 10:44:12.317686   7220.1         :      !! - Thread stack (from mqe.dll)
00000002 10:44:12.322152   7220.1         :      !! - -> MQCONNX
00000003 10:44:12.322641   7220.1         :      !! - -> trmzstMQCONNX
00000004 10:44:12.322656   7220.1         :      !! - -> zstMQCONNX
00000005 10:44:12.322662   7220.1         :      !! - -> zstMQConnect
00000006 10:44:12.322667   7220.1         :      !! - -> zstInitCS
00000007 10:44:12.325228   7220.1         :      !! - -> xcsInitialize
```

Sample excerpt of a Windows trace file for **amqsget**:

```
Process : C:\Program Files\IBM\MQ\Tools\c\Samples\Bin64\amqsget.exe (64-bit)
Arguments :
Host : IBM
Operating System : Windows 7 Professional x64 Edition, Build 7601: SP1
Product Long Name : IBM MQ for Windows (x64 platform)
Version : 9.0.0.0    Level : p900-L160512.4
O/S Registered : 1
Data Path : C:\ProgramData\IBM\MQ
Installation Path : C:\CourseDev\MQV9Install
Installation Name : Installation1 (1)
License Type : Production
UTC Date : 2016/10/25: Time : 14:45:1.580
LCL Date : 2016/10/25: Time : 10:45:1.580  Eastern Standard Time
QueueManager : QM01

Counter  TimeStamp          PID.TID    Ident        Data
============================================================

000006EF 10:45:01.565102   7200.1          :      !! - Thread stack (from mqe.dll)
000006F0 10:45:01.565113   7200.1          :      !! - -> MQCONNX
000006F1 10:45:01.565119   7200.1          :      !! - -> trmzstMQCONNX
000006F2 10:45:01.565124   7200.1          :      !! - -> zstMQCONNX
000006F3 10:45:01.565128   7200.1          :      !! - -> zstMQConnect
000006F4 10:45:01.565131   7200.1          :      !! - -> zstInitCS
000006F5 10:45:01.565136   7200.1          :      !! - -> xcsInitialize
```

End of exercise

Exercise review and wrap-up

Having completed this exercise, you should be able to:

- Set up an IBM MQ trace
- Analyze the output from a trace

Exercise 8. Controlling access to IBM MQ

Estimated time

01;00

Overview

In this exercise, you use the IBM MQ OAM commands to set access control on a queue, and then use the IBM MQ sample programs to see the effect of attempting to breach security.

Objectives

After completing this exercise, you should be able to:

- Use the **setmqaut** command to define access control on a queue
- Use the **dspmqaut** command to display the access control on a queue
- Use IBM MQ Explorer to manage authority records
- Enable and monitor authority events
- Test security by using IBM MQ sample programs

Introduction

In this exercise you use a non-privileged user ID of “oamlabuser” that is a member or the “oamlab” group to access MQ objects, experience access failures, examine related diagnostic messages, and apply the correct rules. You use the MQ Explorer and the **`setmqaut`** command to apply OAM rules.

Linux

All authorities are set at the group level by default. In Exercise 1, you created the queue manager so that you can set authorities at user-level authority by entering the **`-oa user`** option on the **`create queue manager`** command.

In Part 2 of this exercise, you create a generic profile. A generic profile associates with more than one object of the same type. You can grant authorities to a set of objects at the same time by creating an authority record against the generic profile.

Requirements

- IBM MQ and IBM MQ Explorer

- A text editor
- The queue manager QM01 and queues that are created in Exercise 1
- A user that does not belong to the “mqm” group or the “users” group (on Linux)

Exercise instructions

Some steps in this exercise require that you run MQ sample programs to put, get, and browse messages as an unauthorized user that is not a member of the **mqm** group access is required. Other steps require that you complete steps as the Administrator user.

Windows

On the Windows image, the unauthorized user is **oamlabuser**. This user is assigned to the **oamlab** group. It is not a member of the **mqm** or **Administrators** group.

The Administrator user ID is **Administrator**.

Linux

On the Linux image, the unauthorized user is **oamlabuser**. It is not a member of the **mqm** group.

The Administrator user ID is **localuser**.

Part 1: Authorizing a user

In this part of the exercise, you log in as a user **oamlabuser** (the user that does not have object authority) and verify that this user cannot access a queue. The unauthorized user receives a 2035 error when an attempt is made to access the queue from a sample application.

__ 1. If it is not running, start queue manager QM01.

__ 2. Verify that queue QL.A on QM01 is not PUT inhibited.

To display the queue properties in MQSC, type:

```
DIS QL(QL.A)
```

To display the queue properties in MQ Explorer, right-click the queue in the **Queues** view and then click **Properties**.

__ 3. Enable authority events on your queue manager. This method is one way to monitor authority violations.

__ a. Using MQ Explorer, right-click **QM01** in the navigator and click **Properties**.

__ b. On the **Events** tab, select **Enabled** for **Authority events**.

__ c. Click **Apply** and then click **OK**.

__ 4. Open a new command window and start a session (window) as the unauthorized user.

Windows

To start a command window as **oamlabuser**, enter the following command from a Windows command prompt:

```
runas /user:oamlabuser cmd
```

The password is: **passw0rd**

Linux

Open a new terminal window and set the user to **oamlabuser**:

```
su -l oamlabuser
```

The password is: **passw0rd123**

__ 5. In the unauthorized user (**oamlabuser**) command window, run the **amqsput** sample program to attempt to put a message to queue QL.A on the queue manager QM01. Type:

```
amqsput QL.A QM01
```

You should receive an error message with a reason code 2035 (MQRC_NOT_AUTHORIZED)

__ 6. Use the **amqsevt** command to display the queue manager events.

In the administrator command window, type:

```
amqsevt -m QM01 -q SYSTEM.ADMIN.QMGR.EVENT
```

The SYSTEM.ADMIN.QMGR.EVENT queue should include a message similar to the following message:

```
Event Type               : Queue Mgr Event [44]
Reason:                  : Not Authorized [2035]
Event created            : 2016/10/26 09:52_04.54 GMT
   Queue Mgr Name        : QM01
   Reason Qualifier      : Conn Not Authorized
   User Identifier       : oamlabuser
   Appl Type             : Unix
   Appl Name             : amqsput
```

This program continues to run until you either close the window or end the program by typing Ctrl+C.

__ 7. A record of the authority violation is also included in the queue manager **errors** subdirectory.

Locate the error file and open it in a text editor. Scroll to the end of the file to find the most recent entry.

Windows

For Windows queue managers, a security violation message is written to the **`AMQERR01.LOG`** file in the **`C:\ProgramData\IBM\MQ\qmgrs\QM01\errors`** directory.

Linux

For Linux queue managers, a security violation message is written to the **`AMQERR01.LOG`** file in the**`/var/mqm/qmgrs/QM01/errors`** directory.

Error message example:

```
10/26/2016 07:14:41 - Process(2920.12) User(MUSR_MQADMIN) Program(amqzlaa0.exe)
                      Host(WS2008R2X64) Installation(Installation1)
                        VRMF(9.0.0.0) QMgr(QM01)

   AMQ8077: Entity 'oamlabuser@ws2012r2x64' has insufficient authority to
   access object'QM01'.

   EXPLANATION:
   The specified entity is not authorized to access the required object. The
   following requested permissions are unauthorized: connect

   ACTION:
   Ensure that the correct level of authority has been set for this entity
   against the required object, or ensure that the entity is a member of a
   privileged group.
```

__ 8. Use MQ Explorer or the **`setmqaut`** command in the **Administrator** command window to add connection authority for the user **oamlabuser** to the queue manager QM01.

If you use the **`setmqaut`** command, use the **dspmqaut** command to verify that the user has connect authority.

To use the **`setmqaut`** command to add connection authority for the user **oamlabuser**, type:

```
setmqaut -m QM01 –t qmgr -p oamlabuser +connect
```

To use the **dspmqaut** command to verify that the user has connect authority, type:

```
dspmqaut -m QM01 –t qmgr -p oamlabuser
```

To use MQ Explorer to add connection authority for the user **oamlabuser**:

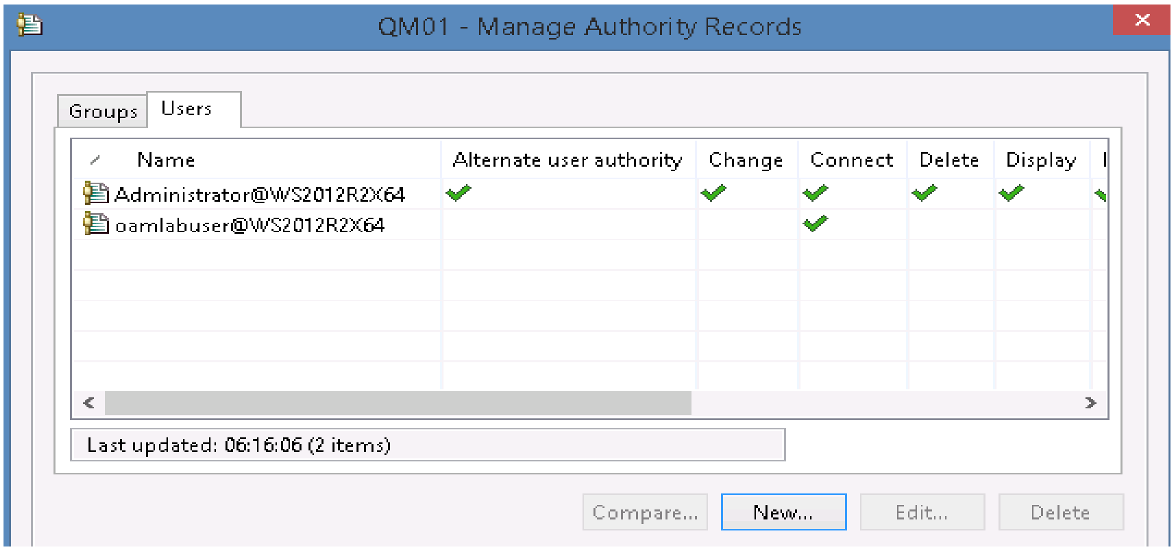
__ a. Right-click the queue manager QM01 in the **MQ Explorer - Navigator** view and then click **Object Authorities > Manage Queue Manager Authority Records**.

__ b. On the **Users** tab, click **New**. Set the **Entity name** to: **`oamlabuser`**

__ c. Select **Connect** under the **MQI** section. Click **OK**.

__ d. Click **OK** on the confirmation window.

__ e. Verify that the user has **Connect authority**, and then click **Close**.



Name	Alternate user authority	Change	Connect	Delete	Display
Administrator@WS2012R2X64	✓	✓	✓	✓	✓
oamlabuser@WS2012R2X64			✓		

Last updated: 06:16:06 (2 items)

__ 9. As the **Administrator**, give the user **oamlabuser** general API access (put, get, browse, inquire, and set) to QL.A on your queue manager.

To use the **setmqaut** command, type:

```
setmqaut -m QM01 -t q -n QL.A -p oamlabuser +put +get +browse +inq +set
```

To use MQ Explorer:

__ a. Right-click the **Queues** folder under QM01 in the **MQ Explorer - Navigator** view and then click **Object Authorities > Manage Authority Records**.

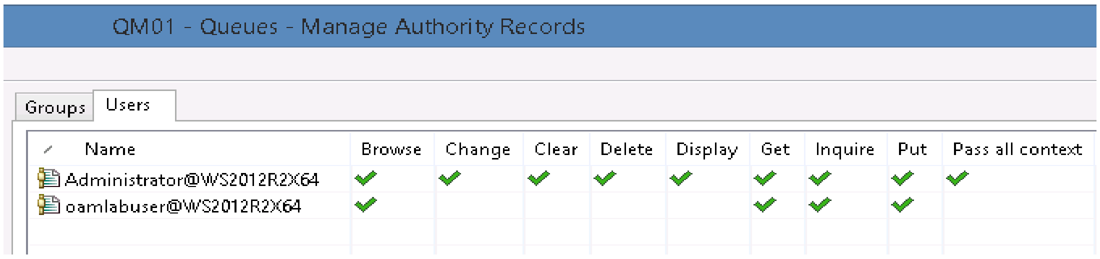
__ b. Expand **Specific Profiles**. Select **QL.A**.

__ c. On the **Users** tab, click **New** and then type **oamlabuser** as the **Entity Name**.

__ d. Select **Browse**, **Get**, **Inquire**, **Put**, and **Set** authority under the **MQI** section. Click **OK**.

__ e. Click **OK** on the confirmation window.

__ f. Verity that the user has **Browse**, **Get**, **Inquire**, **Put**, and **Set** authorities, and then click **Close**.



Name	Browse	Change	Clear	Delete	Display	Get	Inquire	Put	Pass all context
Administrator@WS2012R2X64	✓	✓	✓	✓	✓	✓	✓	✓	✓
oamlabuser@WS2012R2X64	✓					✓	✓	✓	

__ 10. In the command window that is running as **oamlabuser**, put a message on QL.A. This user should now be able to put messages to QL.A.

Type: **`amqsput QL.A QM01`**

__ 11. In the command window that is running as **oamlabuser**, use the **amqsbcg** sample program on QL.A to browse the queue and verify that the access is now allowed.

Type: **`amqsbcg QL.A QM01`**

Part 2: Using generic profiles

In this part of the exercise, you create a generic profile to grant authorities to a set of objects at the same time by creating an authority record against the generic profile.

__ 1. As **Administrator**, use the **`setmqaut`** command to remove the specific profile that allows user **oamlabuser** to access QL.A.

Type: **`setmqaut -m QM01 -t q -n QL.A -p oamlabuser -remove`**

__ 2. As **Administrator**, create a new generic profile that is called **`QL.*.TEST`** that allows the user **oamlabuser** to access (browse, put, get, set, and inquire) queues that match the generic queue name (**`-n`**).

Type: **`setmqaut -m QM01 -t q -n QL.*.TEST -p oamlabuser +allmqi`**

__ 3. As **Administrator**, define a queue on QM01 that is named QL.A.TEST.

Type:

```
runmqsc QM01
DEFINE QL(QL.A.TEST)
END
```

__ 4. As **oamlabuser**, try to access QL.A. You should get the 2035 error.

Type: **`amqsput QL.A QM01`**

__ 5. As **oamlabuser**, try to put a message to QL.A.TEST.

Type: **`amqsput QL.A.TEST QM01`**

This user should be able to put a message to QL.A.TEST because it matches the generic queue name of QL*.TEST.

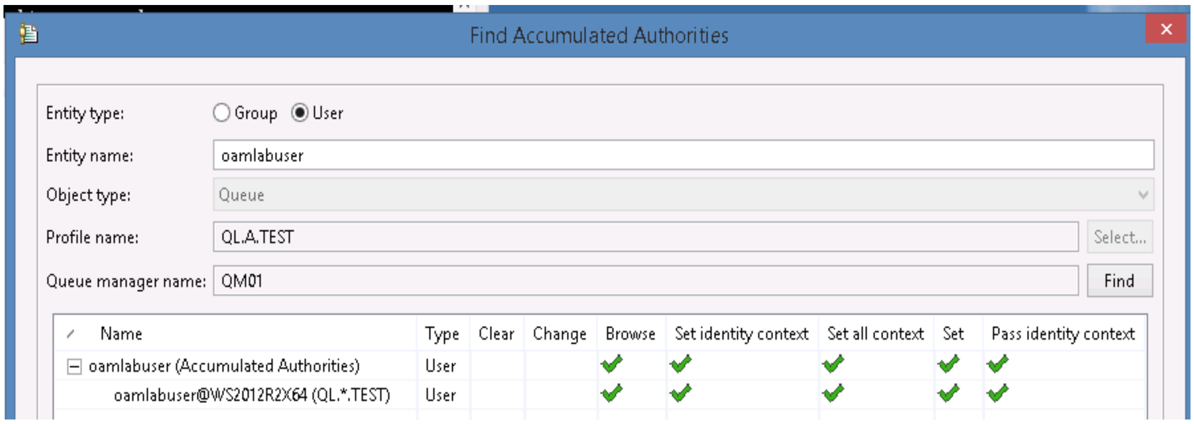
__ 6. As the **Administrator**, use MQ Explorer or the **`dmpmqaut`** command to inspect the profiles that apply to QL.A.TEST queue.

To use MQ Explorer to inspect the profiles that apply to QL.A.TEST, complete the following steps:

__ a. Right-click the queue **QL.A.TEST** in the **Queues** content view and then click **Object Authorities > Find Accumulated Authorities**.

__ b. Click **User** for the Entity Type and then type **`oamlabuser`** for the **Entity name**.

__ c. Click **Refresh**.


Find Accumulated Authorities

Entity type: Group [✓ User]

Entity name: oamlabuser

Object type: Queue

Profile name: QL.A.TEST

Queue manager name: QM01

Name	Type	Clear	Change	Browse	Set identity context	Set all context	Set	Pass identity context
oamlabuser (Accumulated Authorities)	User			✓	✓	✓	✓	✓
oamlabuser@WS2012R2X64 (QL.*.TEST)	User			✓	✓	✓	✓	✓

To use to the **dmpmqaut** command to inspect the profiles that apply to QL.A.TEST, type:

```
dmpmqaut -m QM01 -t q -p oamlabuser -n QL.A.TEST
```

It returns a summary of the profile. For example,

```
profile: QL.*.TEST
object type: queue
entity: oamlabuser@WS2012R2X64(QL.*.TEST)
entity type: principal
authority: allmqi
```

__ 7. Close the command window that is running as the user **oamlabuser**.

Exercise cleanup

__ 1. Disable authority events on the queue manager QM01.

__ a. Using MQ Explorer, right-click **QM01** in the navigator and click **Properties**.

__ b. On the **Events** tab, select **Disabled** for **Authority events**.

__ c. Click **Apply** and then click **OK**.

End of exercise

Exercise review and wrap-up

In the lab environment, you ran as an unauthorized user and enabled authorization to the queue manager and queues.

Having completed this exercise, you should be able to:

- Use the **`setmqaut`** command or MQ Explorer to create authority records for a queue
- Use the **`dspmqaut`** or MQ Explorer to display the authority records for a queue
- Use IBM MQ utilities and sample programs to test security

Exercise 9. Using a media image to restore a queue

Estimated time

00:30

Overview

In this exercise, you capture a media image of a queue, deliberately damage the queue, and then restore it.

Objectives

After completing this exercise, you should be able to:

- Capture an object media image
- Re-create an IBM MQ object from an object media image

Introduction

This exercise demonstrates the survival of persistent messages across a queue manager restart. It also demonstrates how a damaged queue can be recovered from its media image.

In this exercise, you create a queue manager with a linear log. To demonstrate media recovery, you then damage a queue. The act of deliberately damaging a queue should not be done in practice. It is done in this exercise to simulate a disk failure.

You can use the **amqsbcg** sample program or MQ Explorer to browse queues.

Requirements

- IBM MQ and MQ Explorer is installed on the local system

Exercise instructions

Part 1: Queue manager restart

In this part of the exercise, you create a queue manager that uses linear logging. This part of the exercise also demonstrates the survival of persistent messages across a queue manager restart.

__ 1. To avoid any conflicts or problems, use MQ Explorer or the **endmqm** command to stop all running queue managers.

To use the **endmqm** command, type:

```
endmqm -i QM01
endmqm -i QMR01
```

__ 2. Create a queue manager that is named QML01 that uses linear logging, runs on port 1417, and uses the SYSTEM.DEAD.LETTER.QUEUE for its dead-letter queue.

Type: **crtmqm -ll -p 1417 -u SYSTEM.DEAD.LETTER.QUEUE QML01**

__ 3. Start the queue manager.

Type: **strmqm QML01**

__ 4. Use a text editor to create an MQSC command file that is named **recover.mqsc** that defines the following objects.

__ a. Define two local queues that are named QL.A and QL.B. Set the default persistence of each queue to YES (persistent).

In the command file, type:

```
DEF QL(QL.A) DEFPSIST(YES) REPLACE
DEF QL(QL.B) DEFPSIST(YES) REPLACE
```

__ b. Define two alias queues that are named QA.A and QA.B.

Define QA.A so that it resolves to the local queue QL.A.

Define QA.B so that it resolves to the local queue QL.B.

Set the default persistence of each alias queue to NO (not persistent).

In the command file, type:

```
DEF QA(QA.A) TARGET(QL.A) DEFPSIST(NO) REPLACE
DEF QA(QA.B) TARGET(QL.B) DEFPSIST(NO) REPLACE
```

__ 5. Use MQSC to process the command file and redirect the output to a report file.

Make sure that you specify the queue manager name QML01 in the **runmqsc** command. Also, if the current directory does not contain the **recover.mqsc** file, you must specify the path for the **recover.mqsc** file.

Type: **runmqsc QML01 < recover.mqsc > recoverreport.txt**

Review the report file to verify that all commands in the file ran successfully.

__ 6. Use the **amqsput** sample program to put persistent and non-persistent messages on the local queues QL.A and QL.B and alias queues QA.A and QA.B.

When the **amqsput** sample program starts, the value of the **DefPersistence** attribute of the queue determines whether the message that is put on the queue is persistent or non-persistent. Because of the queue definitions that you made in step 4, the **amqsput** sample program puts persistent messages on the local queues and non-persistent messages on the alias queues.

Type:

```
amqsput QL.A QML01
amqsput QA.A QML01
amqsput QL.B QML01
amqsput QA.B QML01
```

__ 7. Use the MQ Explorer or the sample program **amqsbcg** to browse the messages on the local queue QL.A and QL.B and verify that it contains a mixture of persistent and non-persistent messages.

To use the sample program, type: **`amqsbcg QL.A QML01`**

The sample program returns a record for each message that is on the queue.

- If the message is persistent, the **Persistence** attribute is **`1`**.
- If the message is not persistent, the **Persistence** attribute is **`0`**.

To use MQ Explorer, right-click the queue in the **Queues** content view and then click **Browse Messages**. A row exists for each message on the queue. Scroll to the right until you see the **Persistence** column and verify that the queue contains both persistent and non-persistent messages.

__ 8. Stop and then restart the queue manager QML01.

Type:

```
endmqm -i QML01
strmqm QML01
```

__ 9. Use MQ Explorer or the sample program **amqsbcg** to browse the messages on QL.A and QL.B again. Verify that only the persistent messages on the local queue survived the restart.

Part 2: Media recovery

__ 1. Run the **rcdmqimg** command against the queue QL.B to write an image of the queue to the log for use in media recovery.

Type: **`rcdmqimg -m QML01 -t ql QL.B`**

__ 2. Locate the file that implements local queue QL.B within the file system and then damage the queue by deleting the file. The file location is specific to the operating system.

Linux

```
rm /var/mqm/qmgrs/QML01/queues/QL\!B
```

Windows

```
erase "C:\ProgramData\IBM\MQ\qmgrs\QML01\queues\QL!B"
```

__ 3. Use the **`DISPLAY QUEUE`** command in **`runmqsc`** to display the attributes of local queue QL.B. This command still works because it does not yet recognize that the queue is damaged.

Type:

```
runmqsc QML01
DIS Q(QL.B)
END
```

__ 4. Use **`amqsput`** to put some messages to local queue QL.B. It might take 10 or more messages (because of buffering) but the queue manager eventually detects that the queue is damaged and reports the damage by returning MQRC 2101 (MQRC_OBJECT_DAMAGED).

Type: **`amqsput QL.B QML01`**

__ 5. Use the **`DISPLAY QUEUE`** command in **`runmqsc`** to display the attributes of local queue QL.B.

Type:

```
runmqsc QML01
DIS Q(QL.B)
end
```

The queue manager now recognizes that a queue is damaged and reports that the MQ object QL.B is not found.

__ 6. Use the re-create object (**`rcrmqobj`**) command to recover local queue QL.B from its media image.

Type: **`rcrmqobj -m QML01 -t ql QL.B`**

You should see a message that indicates that the local queue QL.B was re-created.

__ 7. In **`runmqsc`**, check that you can now display the attributes of the queue. Type:

```
runmqsc QML01
DIS Q(QL.B)
end
```

__ 8. Use MQ Explorer or the **`amqsbcg`** sample program to browse the local queue QL.B to check whether the messages are recovered successfully.

To use the sample program to browse the queue, type: **`amqsbcg QL.B QML01`**

To browse the messages by using MQ Explorer, right-click the QL.B queue in the **Queues** content view and then click **Browse Messages**.

Were the non-persistent messages recovered?

Exercise cleanup

__ 1. Clear the messages from QL.A and QL.B by using a sample program or MQ Explorer.

To clear the messages by using a sample program, type: **`amqsget QL.B QML01`**

To clear the messages by using MQ Explorer, right-click the queue in the **Queues** content view, click **Clear Messages**, and then click **Clear**.

__ 2. Stop and then delete the queue manager QML01. It might take 1 or 2 minutes for the queue manager to be deleted.

Type:

```
endmqm -i QML01
dltmqm QML01
```

End of exercise

Exercise review and wrap-up

Having completed this exercise, you should be able to:

- Create a queue manager with linear logging
- Put persistent and non-persistent messages on local queues
- Stop and restart a queue manager to verify that persistent messages are maintained during a queue manager restart
- Record a media image
- Recover a damaged local queue from its media image

Exercise 10. Backing up and restoring IBM MQ object definitions

Estimated time

00:30

Overview

In this exercise, you use the **dmpmqcfg** command to unload a queue manager's object definitions. You then create a queue manager and load the same definitions, and use MQSC commands or IBM MQ Explorer to show that the definitions are the same.

Objectives

After completing this exercise, you should be able to:

- Use IBM MQ commands to back up object definitions of a queue manager
- Use IBM MQ commands to upload object definitions to another queue manager

Introduction

The IBM MQ **`dmpmqcfg`** control command can be used to back up and restore a queue manager’s configuration. As an option, you can use the **`-o setmqaut`** parameter to create **`setmqaut`** commands that can be used to re-create the queue manager security definitions.

Requirements

- IBM MQ and IBM MQ Explorer
- The queue manager QM01 that was created in Exercise 1

Exercise instructions

Part 1: Backing up queue manager configuration

__ 1. Display the status of the queue manager QM01 to verify that it is running.

In a command window, type: **`dspmq -m QM01`**

If the queue manager is not running, start it now.

__ 2. Run the MQ **`dmpmqcfg`** command to back up QM01.

Use the default formatting option of MQSC and all attributes. Use standard output redirection to store the definitions into a file.

```
dmpmqcfg -m QM01 -a > QM01.mqsc
```

__ 3. Using a text editor, examine the resulting file, and answer the following questions.

__ a. Verify that SYSTEM.* objects appear in the backup file. Can you suppress generation of the SYSTEM.* objects?

__ b. Is the REPLACE option present in the generated definitions?

__ c. What does REPLACE do?

Part 2: Restoring the object definitions

In this part of the exercise, you simulate the running of the generated backup file. This action would be done if you needed to redefine all your MQ objects. In this exercise, you are not replacing any definitions because some of the SYSTEM queues are not empty. Redefining the SYSTEM queue objects effectively removes all cluster, channel state, and authorities that are stored in the SYSTEM queues.

__ 1. Use MQSC to check whether any of the SYSTEM queues on QM01 contain messages.

In MQSC, type:

```
DIS QL(SYSTEM.*) WHERE(CURDEPTH GT 0)
```

__ 2. Use the **`runmqsc`** command with the **`-v`** option to run in verify mode and simulate the restore function without replacing the definitions. Redirect the output to a file so that you can verify the success or failure of each command.

```
runmqsc -v QM01 < QM01.mqsc > QM01.out
```

Important

Ensure that you use the **`-v`** option so that you do not overwrite the existing configuration in the lab environment.

__ 3. Use a text editor to examine the output in **`QM01.out`** file that you created in Step 2.

Were there any errors? If yes, find and correct the error, and then rerun the command with the **`-v`** switch.

__ 4. Use an MQSC command to verify that messages on the SYSTEM queues on QM01 are intact.

In MQSC, type:

```
DIS QL(SYSTEM.*) WHERE(CURDEPTH GT 0)
```

Ensure that the queue depths match the queue depths reported in Part 2, Step 1.

Part 3: Dumping security definitions

In this part of the exercise, you use an MQ command and the MQ Explorer to write security definitions to a file.

__ 1. Use the MQ **`dmpmqcfg`** command with the **`-x authrec`** option to write the queue manager security definitions to a file. Use the **`-o setmqaut`** option to format the output as a series of **`setmqaut`** commands.

```
dmpmqcfg -o setmqaut -x authrec -t qmgr -m QM01 > QM01-setmqaut.mqsc
```

__ 2. Use MQ Explorer to write the OAM definitions for QM01.

__ a. In the MQ Explorer Navigator, right-click QM01 and then click **Object Authorities > Save All**.

__ b. On Windows, select a directory, enter a file name, and then click **Save**.

On Linux, select a directory, enter a file name, and then click **OK**.

__ 3. Examine the output from the security definitions file.

Find the **`setmqaut`** entries for QL.A.

__ 4. Run MQ command: **`dmpmqaut -m QM01 -n QL.A -t q`**

This command writes a formatted view of the authorities for the queue QL.A.

__ 5. Examine the output from the **`dmpmqaut`** command and compare it with the output from the **`dmpmqcfg`** command.

The **`dmpmqcfg`** command writes the security definitions as a list of **`setmqaut`** commands. These commands can then be used to re-create the security definitions on a new queue manager.

In contrast, **`dmpmqaut`** writes a formatted view of the authorities. The formatted view is useful for finding all the authorities that are set against a particular MQ object. The output cannot be used to re-create the security definitions without reformatting.

End of exercise

Exercise review and wrap-up

Having completed this exercise, you should be able to:

- Use IBM MQ commands to back up object definitions of a queue manager
- Use IBM MQ commands to upload object definitions to another queue manager

Exercise solutions

Part 1, Step 3: Using a text editor, examine the resulting file, and answer the following questions.

a. Verify that SYSTEM.* objects appear in the backup file. Can you suppress generation of the SYSTEM.* objects?
   Answer: **No**.

b. Is the REPLACE option present in the generated definitions?
   Answer: **Yes**.

c. What does REPLACE do?
   Answer: **The REPLACE and NOREPLACE options control whether any existing definitions are replaced with the new ones. The MQ default is NOREPLACE**.

Part 2 and Part 3: Sample solution files are provided in the lab files **`Lab10\solution`** subdirectory.

Exercise 11. Implementing a basic cluster

Estimated time

01:00

Overview

In this exercise, you use IBM MQ Explorer to create a cluster of four queue managers. You then test the cluster by using the cluster mechanism to send messages between queues on all queue managers in the cluster.

Objectives

After completing this exercise, you should be able to:

- Use IBM MQ Explorer to define a simple queue manager cluster
- Use the IBM MQ sample programs to test a cluster environment

Introduction

In this exercise you use IBM MQ Explorer to create a basic cluster that is named CLUS1 with four queue managers:

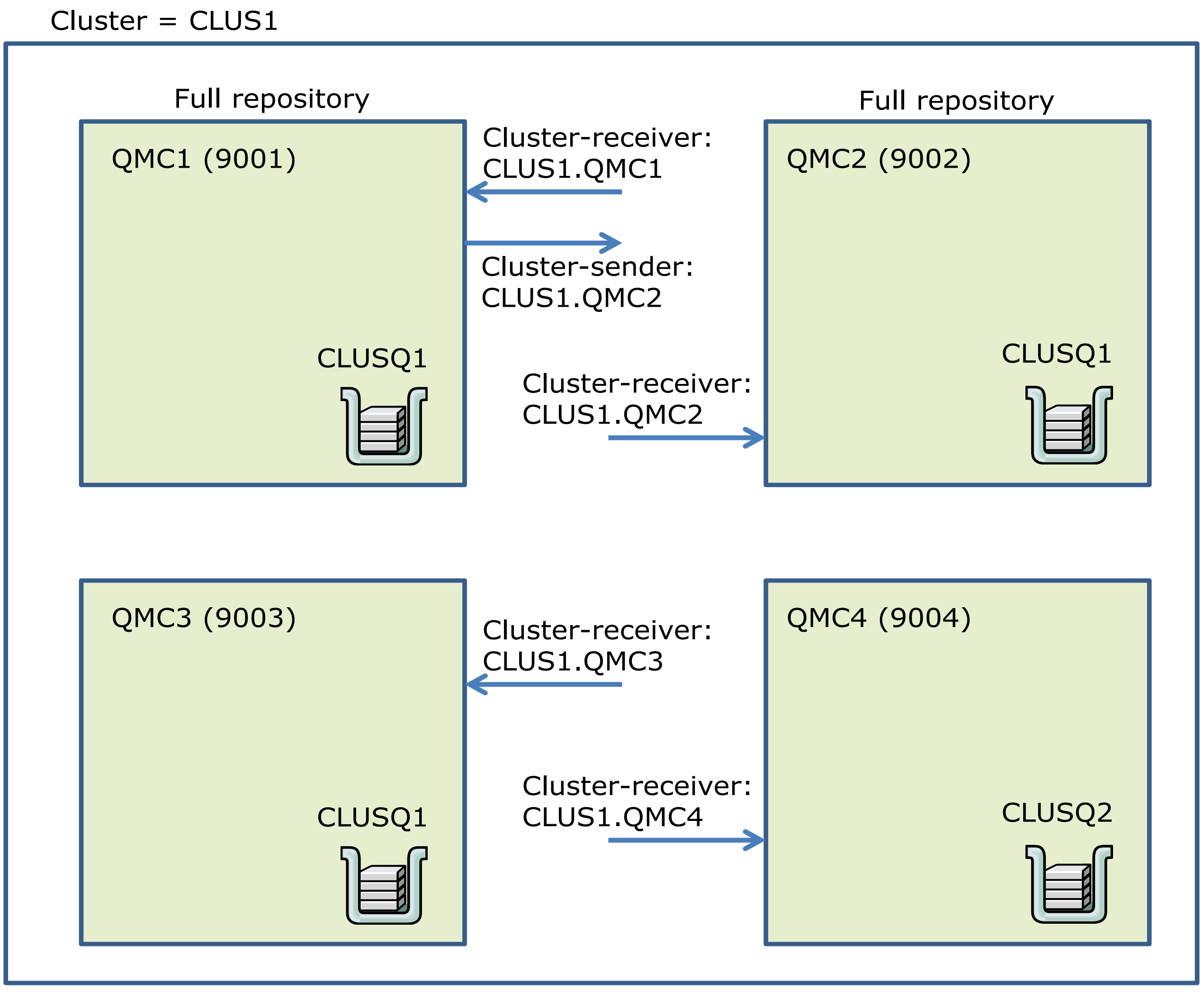
- Full repository queue manager QMC1 on localhost(9001)
- Full repository queue manager QMC2 on localhost(9002)
- QMC3 on localhost(9003)
- QMC4 on localhost(9004)

You also define the following channel definitions:

- One cluster-receiver (CLUSRCVR) channel to each queue manager in the cluster
- One cluster-sender (CLUSSDR) channel between the full repository queue managers, QMC1 and QMC2

After you create the queue managers and the cluster, you define local queues that are used for cluster queues as follows:

- CLUSQ1 on QMC1, QMC2, QMC3
- CLUSQ2 on QMC4



Cluster = CLUS1
Full repository
Full repository
QMC1 (9001)
Cluster-receiver: CLUS1.QMC1
QMC2 (9002)
Cluster-sender: CLUS1.QMC2
CLUSQ1
Cluster-receiver: CLUS1.QMC2
CLUSQ1
QMC3 (9003)
Cluster-receiver: CLUS1.QMC3
QMC4 (9004)
Cluster-receiver: CLUS1.QMC4
CLUSQ1
CLUSQ2

Requirements

- IBM MQ and IBM MQ Explorer
- **`data.txt`** file in the **`C:\labfiles\Lab11\data`** directory (on Windows) or **`/home/localuser/labfiles/Lab11/data`** directory (on Linux)

Exercise instructions

Part 1: Defining the cluster queue managers, channels, and queues

In this part of the exercise, you use IBM MQ Explorer to define the cluster queue managers, channels, and clustered queues. You also use IBM MQ Explorer to verify your configuration.

__ 1. If it is not already running, start IBM MQ Explorer.

__ 2. Stop any existing queue managers that are running.

__ 3. Using IBM MQ Explorer, create the following queue managers as described in the table. Accept the default values for any properties that are not listed in the table.

Queue manager name	**Listener port number**	**Dead-letter queue**
QMC1	9001	SYSTEM.DEAD.LETTER.QUEUE
QMC2	9002	SYSTEM.DEAD.LETTER.QUEUE
QMC3	9003	SYSTEM.DEAD.LETTER.QUEUE
QMC4	9004	SYSTEM.DEAD.LETTER.QUEUE

__ 4. Create the queue manager cluster.

__ a. In the **IBM MQ Explorer - Navigator** view, right-click **Queue Manager Clusters** and then click **New > Queue manager cluster.**

__ b. For the cluster name, enter **`CLUS`**1 and then click **Next**.

__ c. For the first full repository queue manager, select **QMC1** and then click **Next**.

Information

The MQSC command to define QMC1 as a full repository queue manager is:

```
ALTER QMGR REPOS(CLUS1)
```

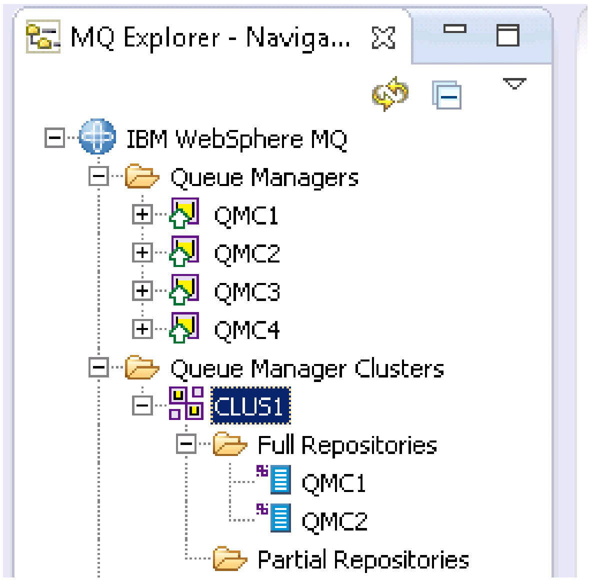
__ d. For the second full repository queue manager, select **QMC2** from the list and then click **Next**.

__ e. Click **Next** to define the cluster channels.

__ f. Change the name of cluster-receiver channel for QMC1 to **`CLUS1.QMC1`** and the connection name to **`localhost(9001)`** and then click **Next**.

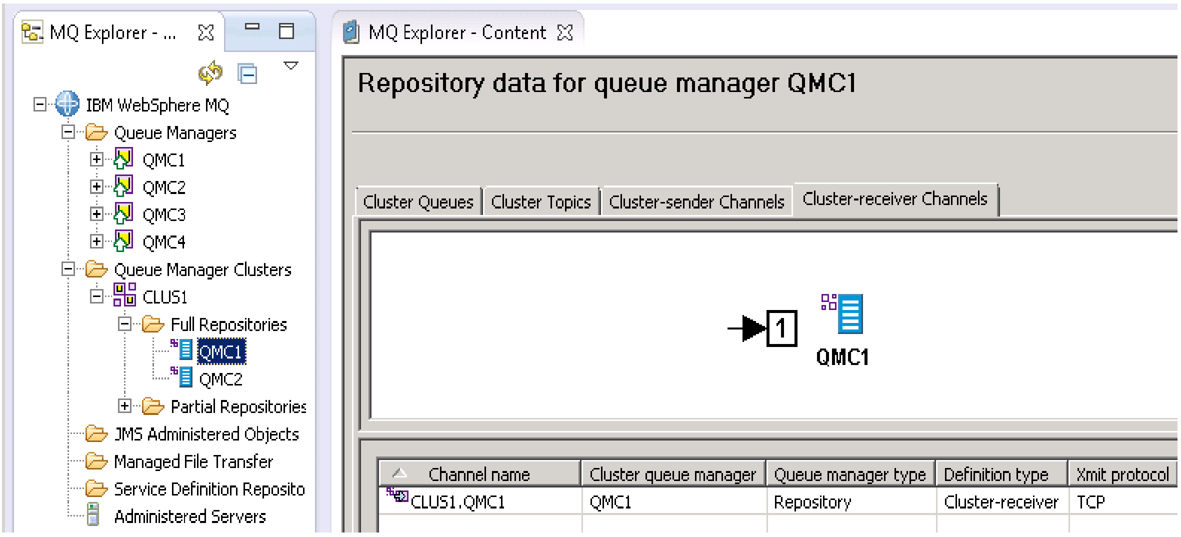
__ g. Change the name of cluster-receiver channel for QMC2 to **`CLUS1.QMC2`** and the connection name to **`localhost(9002)`** and then click **Finish**.

__ 5. Expand the **Queue Managers Clusters** folder in the IBM MQ Explorer and verify that you have a cluster that is named CLUS1 with QMC1 and QMC2 listed as full repositories.



__ 6. Select the full repository queue manager QMC1 under the **Full Repositories** folder.

__ 7. Verify that QMC1 has one cluster-receiver channel that is named CLUS1.QMC1 and one cluster-sender channel that is named CLUS1.QMC2.



Repository data for queue manager QMC1

Channel name	Cluster queue manager	Queue manager type	Definition type	Xmit protocol
CLUS1.QMC1	QMC1	Repository	Cluster-receiver	TCP

__ 8. Use IBM MQ Explorer to add QMC3 to the cluster as partial repositories.

__ a. Right-click **CLUS1** in the **MQ Explorer-Navigator** view and then click **Add Queue Manager to Cluster**.

__ b. Select **QMC3** and then click **Next**.

__ c. Select **Partial repository** and then click **Next**.

__ d. Change the name of the cluster-receiver channel to **`CLUS1.QMC3`** and the connection name to **`localhost(9003)`**. Click **Next**.

__ e. Select **QMC1** as the full repository queue manager and then click **Next**.

__ f. Accept the default to use the cluster-receiver channel **CLUS1.QMC1**. Click **Next**.

__ g. Verify your configuration in the summary and then click **Finish**.

Information

To use MQSC to add a partial repository to a cluster, you would define a cluster-receiver channel for QMC3 and define a cluster-sender channel that points to one of the full repository queue managers.

The equivalent MQSC commands to add QMC3 to the cluster as a partial repository are:

```
DEFINE CHANNEL(CLUS1.QMC3) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) +
CONNAME('localhost(9003)') CLUSTER(CLUS1)

DEFINE CHANNEL(CLUS1.QMC1) CHLTYPE(CLUSSDR) TRPTYPE(TCP) +
CONNAME('localhost(9001)') CLUSTER(CLUS1)
```

__ 9. Use MQ Explorer to add QMC4 to the cluster as a partial repository.

__ a. Right-click **CLUS1** in **MQ Explorer-Navigator** view and then click **Add Queue Manager to Cluster**.

__ b. Select **QMC4** and then click **Next**.

__ c. Select **Partial repository** and then click **Next**.

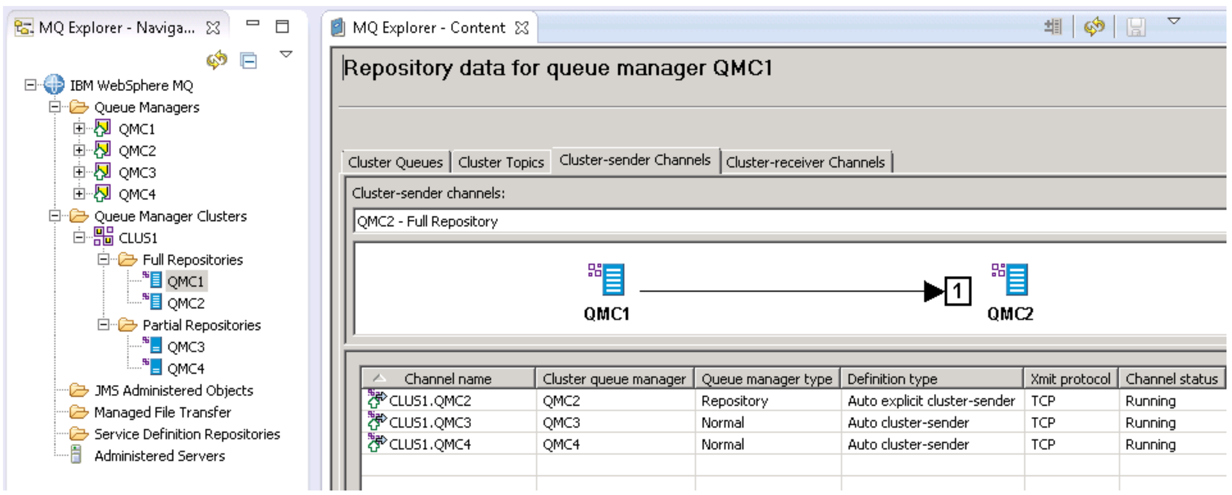
__ d. Create a cluster-receiver channel that is named `CLUS1.QMC4` and the connection `localhost(9004)`.

__ e. Select QMC1 as the full repository queue manager.

__ f. Accept the default to use the cluster-receiver channel **CLUS1.QMC1**. Click **Next**.

__ g. Verify your configuration in the summary and then click **Finish**.

__ 10. Select QMC1 under the **Full Repositories** folder and verify that cluster-sender channels were automatically defined between the partial repositories QMC3 and QMC4 and the full repository QMC1.



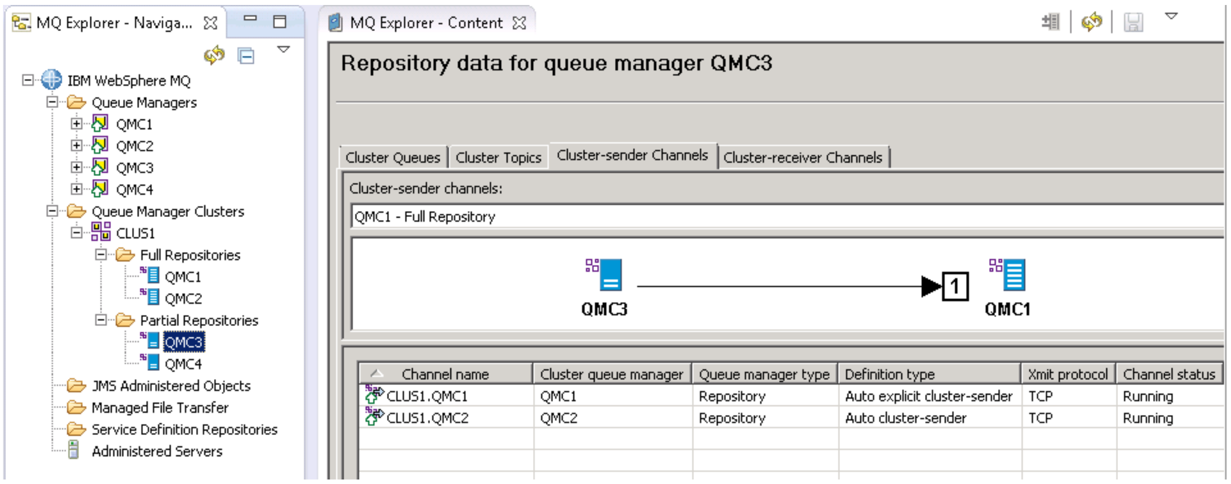
Repository data for queue manager QMC1

Cluster-sender channels: QMC2 - Full Repository

Channel name	Cluster queue manager	Queue manager type	Definition type	Xmit protocol	Channel status
CLUS1.QMC2	QMC2	Repository	Auto explicit cluster-sender	TCP	Running
CLUS1.QMC3	QMC3	Normal	Auto cluster-sender	TCP	Running
CLUS1.QMC4	QMC4	Normal	Auto cluster-sender	TCP	Running

__ 11. Select QMC3 under the **Partial Repositories** folder and verify that QMC3 has cluster-sender channels to both full repositories (QMC1 and QMC2).

Notice that the channel to the second full repository (QMC2) was automatically defined.



Repository data for queue manager QMC3

Cluster-sender channels:

QMC1 - Full Repository

Channel name	Cluster queue manager	Queue manager type	Definition type	Xmit protocol	Channel status
CLUS1.QMC1	QMC1	Repository	Auto explicit cluster-sender	TCP	Running
CLUS1.QMC2	QMC2	Repository	Auto cluster-sender	TCP	Running

__ 12. Create a cluster queue that is named **CLUSQ1** on QMC1.

__ a. Expand QMC1 under the **Queue Managers** folder in the **MQ Explorer - Navigator** view.

__ b. Right-click **Queues** and then click **New > Local Queue**.

__ c. Enter **`CLUSQ1`** for the queue name and then click **Next**.

__ d. On the **Cluster** properties page, click **Shared in cluster** and enter **`CLUS1`** for the cluster name and then click **Finish**.

Information

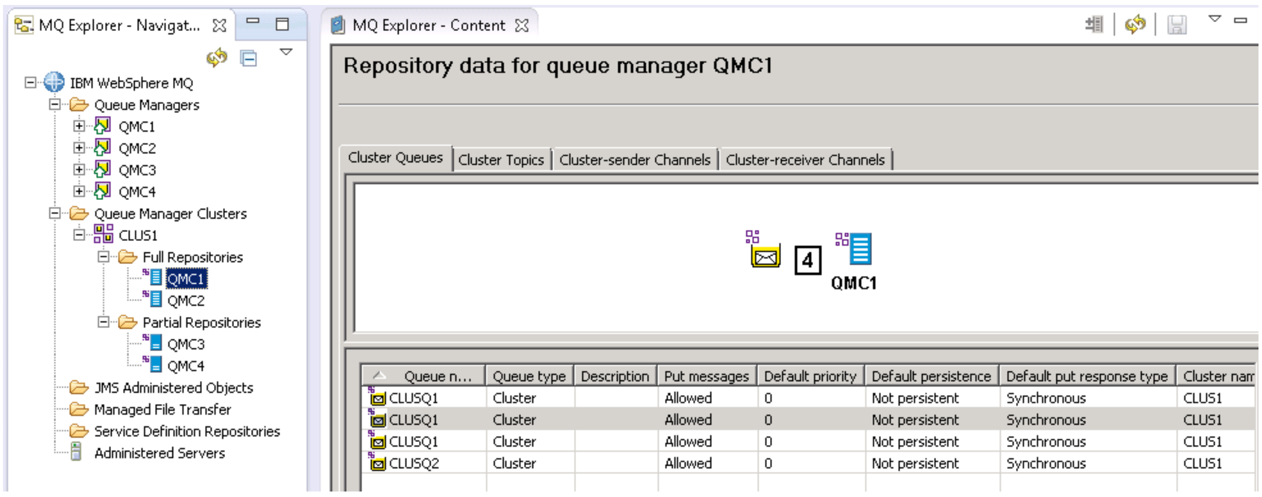
The MQSC command for creating a cluster queue is: **`DEF QL(CLUSQ1) CLUSTER(CLUS1)`**.

__ 13. Follow the procedure in Step 12 and create the following queues on QMC2, QMC3, QMC4:

- On QMC2, create a cluster queue on CLUS1 that is named **`CLUSQ1`**.
- On QMC3, create a cluster queue on CLUS1 that is named **`CLUSQ1`**.
- On QMC4, create a cluster queue on CLUS1 that is named **`CLUSQ2`**.

__ 14. Select **QMC1** under the **Full Repositories** folder and then click the **Cluster Queues** tab to verify that you have a cluster queue on each queue manager.

You should have a cluster queue that is named **CLUSQ1** on QMC1, QMC2, and QM3 and a cluster queue that is named **CLUSQ2** on QMC4.



Repository data for queue manager QMC1

Queue n...	Queue type	Description	Put messages	Default priority	Default persistence	Default put response type	Cluster nam
CLUSQ1	Cluster		Allowed	0	Not persistent	Synchronous	CLUS1
CLUSQ1	Cluster		Allowed	0	Not persistent	Synchronous	CLUS1
CLUSQ1	Cluster		Allowed	0	Not persistent	Synchronous	CLUS1
CLUSQ2	Cluster		Allowed	0	Not persistent	Synchronous	CLUS1

Part 2: Testing the cluster

In this scenario, CLUSQ1 is defined locally on three queue managers: QMC1, QMC2, and QMC3.

In the most basic scenario, by using the default configuration all messages that are put to the cluster queue CLUSQ1 are put on the local manager.

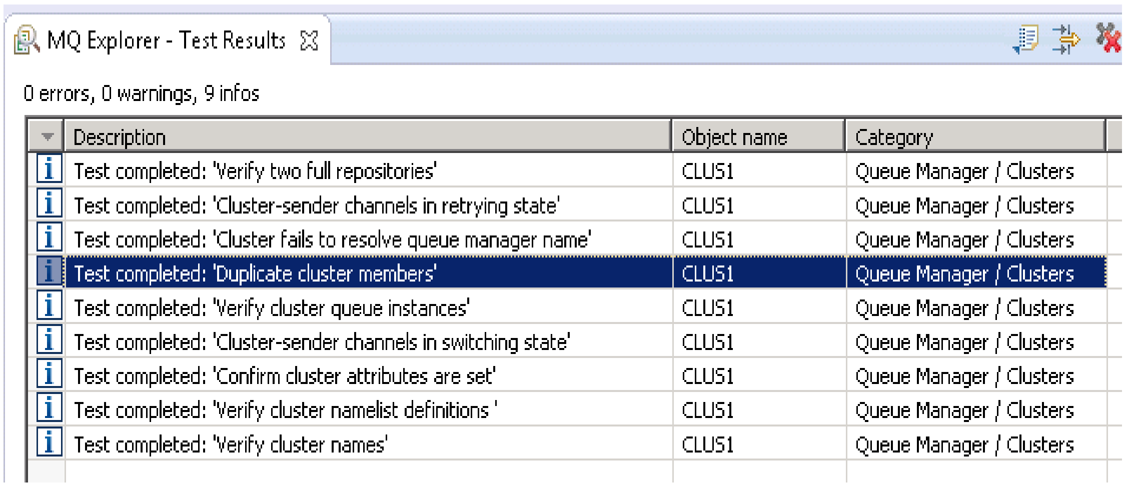
__ 1. Test the cluster object definitions and verify that no errors or potential problems exist.

__ a. In the **MQ Explorer - Navigator** view, right-click **CLUS1** and then click **Test > Run Default Tests**.

__ b. Test results are displayed in the **Test Results** view. The cluster default tests include the following tests:

- **Verify two full repositories**: Checks that all clusters have at least two queue managers that maintain full repositories of the cluster.
- **Cluster-sender channels in retrying state**: Checks whether any of the manually defined cluster-sender channels are still in a *Retrying* state.
- **Cluster fails to resolve queue manager name**: Checks that clusters can successfully resolve all queue manager names.
- **Duplicate cluster members**: Checks whether any cluster memberships list the same queue manager more than one time.
- **Verify cluster queue instances**: Verifies that all instances of a cluster queue have the same attributes.
- **Cluster-sender channels in switching state**: Checks whether any of the manually defined cluster-sender channels are still in a *Switching* state.
- **Confirm that cluster attributes are set**: Checks that all cluster channels have a cluster value.

- **Verify cluster namelist definitions**: Verifies the cluster namelist attributes of queues, channels, and queue managers. The test displays errors if matching namelists cannot be found, or if a namelist is empty.
- **Verify cluster names**: Checks the cluster name attributes of queues, channels, and queue managers. The test checks for names that are similar enough to cause confusion, for example, names that are identical except for capitalization.



0 errors, 0 warnings, 9 infos

Description	Object name	Category
Test completed: 'Verify two full repositories'	CLUS1	Queue Manager / Clusters
Test completed: 'Cluster-sender channels in retrying state'	CLUS1	Queue Manager / Clusters
Test completed: 'Cluster fails to resolve queue manager name'	CLUS1	Queue Manager / Clusters
Test completed: 'Duplicate cluster members'	CLUS1	Queue Manager / Clusters
Test completed: 'Verify cluster queue instances'	CLUS1	Queue Manager / Clusters
Test completed: 'Cluster-sender channels in switching state'	CLUS1	Queue Manager / Clusters
Test completed: 'Confirm cluster attributes are set'	CLUS1	Queue Manager / Clusters
Test completed: 'Verify cluster namelist definitions '	CLUS1	Queue Manager / Clusters
Test completed: 'Verify cluster names'	CLUS1	Queue Manager / Clusters

__ 2. Use the sample program **amqsput** and the supplied text file **data.txt** to write 15 messages to the cluster queue CLUSQ1.

Windows

Type: **amqsput CLUSQ1 QMC1 < C:\labfiles\Lab11\data\data.txt**

Linux

Type: **amqsput CLUSQ1 QMC1 < /home/localuser/labfiles/Lab11/data/data.txt**

__ 3. Use IBM MQ Explorer to view the queue depth of CLUSQ1 on QMC1, QMC2, and QMC3.

What do you notice? You should see that all 15 messages were put to the queue manager QMC1.

The default queue definition has the **Default bind type** (DEFBIND) set to **Open** and the **Cluster workload use queue** (CLWLUSEQ) set to **Queue manager**. These settings explain why all the messages were placed onto one queue manager.

- DEFBIND(OPEN) binds the queue handle to a specific instance of the cluster queue when the queue is opened.
- CLWLUSEQ(QMGR) means that the CLWLUSEQ attribute of the queue manager definition specifies the behavior. By default, it is LOCAL and the target of an MQPUT is the local queue instance, if one exists.

__ 4. Change the queue definition for CLUSQ1 queue on QMC1, QMC2, and QMC3 so that **Default bind type** is set to **Not fixed** and **Cluster workload use queue** is set to **Any**.

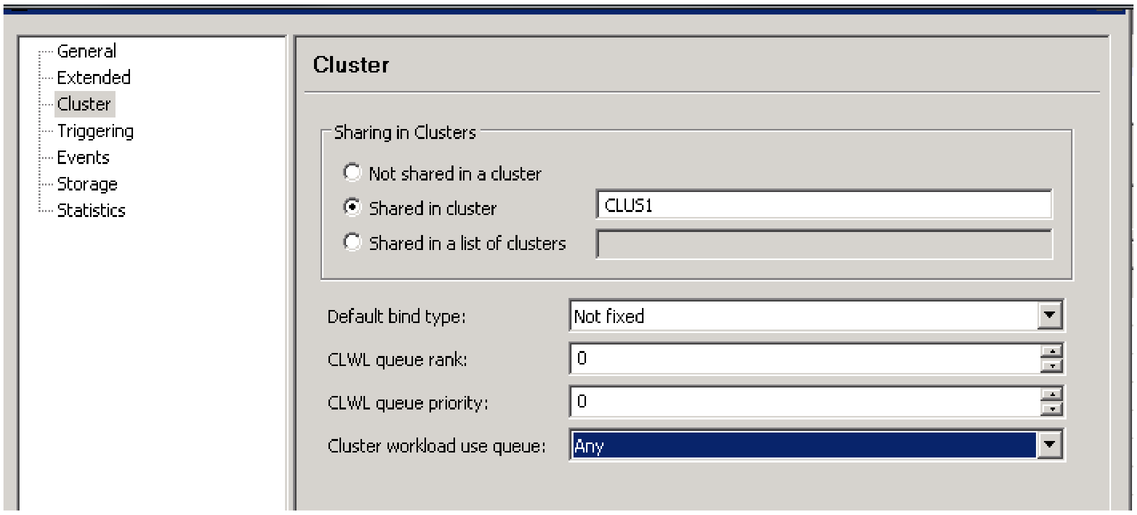
__ a. Right-click **CLUSQ1** on the **Queue** view and then click **Properties**.

__ b. Click **Cluster** to display the **Cluster** properties.

__ c. Change **Default bind type** set to **Not fixed** so that the queue handle is not bound to any one instance of the cluster queue.

__ d. Change **Cluster workload use queue** to **Any** so that any queue can be used.

__ e. Click **Apply** and then click **OK**.



Cluster

Sharing in Clusters

- Not shared in a cluster
- [✓ Shared in cluster] CLUS1
- Shared in a list of clusters

Default bind type: Not fixed

CLWL queue rank: 0

CLWL queue priority: 0

Cluster workload use queue: Any

Information

The MQSC command for changing the cluster queue properties is:

```
ALTER QL(CLUSQ1) DEFBIND(NOTFIXED) CLQLUSEQ(ANY)
```

__ 5. Use IBM MQ Explorer to clear the messages from CLUSQ1 on queue manager QMC1.

__ 6. Repeat Step 2 to use the `amqsput` sample program with the data.txt file to put 15 messages to the cluster queue CLUSQ1.

__ 7. View the queue depth of CLUSQ1 on QMC1, QMC2, and QMC3.

You should see that the messages are now distributed between all three instances of the CLUSQ1 queue on the three queue managers.

Exercise cleanup

Stop the cluster queue managers: QMC1, QMC2, QMC3, and QMC4

End of exercise

Exercise review and wrap-up

In the first part of the exercise, you used MQ Explorer to create a cluster with two full repository queue managers and two partial repository queue managers. You then created cluster queues on each queue manager and tested the cluster.

Exercise 12. Monitoring IBM MQ for performance

Estimated time

01:30

Overview

In this exercise, you enable and configure the online monitoring, statistics, and accounting features of IBM MQ.

Objectives

After completing this exercise, you should be able to:

- Enable accounting and statistics collection in IBM MQ
- View accounting and statistics data
- Configure a queue manager for online monitoring
- Monitor system resource usage

Introduction

With the IBM MQ monitoring facilities, you can postprocess monitoring data and monitor your queue manager online.

This exercise focuses on four types of monitoring:

1. **Statistics monitoring**
   Statistics monitoring generates data about the activity of the queue manager for a specified time interval.

   Statistics monitoring is used to collect information about the performance of various areas of the queue manager. You can collect data on the queue manager, queues, and channels, and you can configure the amount of information to collect.

   After it is collected, the data is made available at regular intervals in the form of PCF records that are written to the SYSTEM.ADMIN.STATISTICS.QUEUE system queue, with one message per subtype.

   This information can be used for charging purposes or for generating historical analysis of the operation or throughput of the IBM MQ system.

2. **Accounting monitoring**
   Accounting monitoring generates data about the activity of an IBM MQ connection.

Accounting monitoring collects information about each IBM MQ connection and when the connection disconnects. This information is written in the form of a PCF monitoring message to the SYSTEM.ADMIN.ACCOUNTING.QUEUE system queue. As an option, Interim messages can be written for long-running tasks.

As with statistics monitoring, the user can configure the amount of information that is collected. Separate messages can be written for each subtype of information that is collected.

This information can be used for charging or accounting purposes. The purpose of the online monitoring is to give a quick view of the performance of a specific resource. Data is immediately available, giving the user an instantaneous view of the particular resource. The monitoring data is collected for channels and queues, and is displayed as extra attributes on the **`DISPLAY CHSTATUS`** and the **`DISPLAY QSTATUS`** MQSC commands.

3. **Online real-time monitoring**
   Some queue and channel status attributes hold monitoring information, if real-time monitoring is enabled. You can enable or disable real-time monitoring for individual queues or channels, or for multiple queues or channels.
4. **System resource monitoring**
   Each queue manager publishes resource usage data to topics. This data is used by subscribers to those topics. When a queue manager starts, the queue manager publishes a set of messages on meta-topics. These messages describe which resource usage topics are supported by the queue manager, and the content of the messages published to those topics. Administrative tools can subscribe to the metadata to discover what resource usage information is available, and on what topics, and then subscribe to the advertised topics.

In this exercise, you configure and use the monitoring functions of IBM MQ. The exercise includes the following tasks:

1. Configure a queue manager to collect statistics data.
2. Configure a queue manager to collect accounting data.
3. Run test programs to generate statistics and accounting data.
4. Use the **amqsmon** sample program to display the information that is contained within accounting and statistics messages in a readable format. The **amqsmon** program reads accounting messages from the accounting queue, SYSTEM.ADMIN.ACCOUNTING.QUEUE, and reads statistics messages from the statistics queue, SYSTEM.ADMIN.STATISTICS.QUEUE.
5. Configure a queue manager for online monitoring and display the monitoring information.
6. Use the **`amqsrua`** command to enable and display MQ resource monitoring data. The **`amqsrua`** command reports metadata that queue managers publish. This data can include information about the CPU, memory, and disk usage.

Requirements

In addition, the following are required:

- IBM MQ and IBM MQ Explorer
- Successful completion of Exercise 4, “Connecting queue managers”
- The **`amqsput`**, **`amqsget`**, **`amqsmon`**, and **`amqsrua`** sample programs that are supplied with IBM MQ

- The **`statsdata.txt`** text file that is provided in the **`C:\labfiles\Lab12\data`** directory on Windows or the **`home/localuser/labfiles/Lab12/data`** directory on Linux

Exercise instructions

Part 1: Configure a queue manager to collect statistics data

In this part of the exercise, you configure a queue manager to collect statistics data. The configuration is controlled by using a set of queue manager and queue/channel attributes.

The attributes can be set by using the MQ Explorer or by using MQSC commands; instructions in this exercise use MQ Explorer.

__ 1. If it is not already running, start MQ Explorer.

__ 2. If it is not already running, start the queue manager that is named QM01.

__ 3. Edit the properties for queue manager QM01 to configure statistics monitoring properties.

   __ a. In the **MQ Explorer - Navigator** view, right-click **QM01** and then click **Properties**.

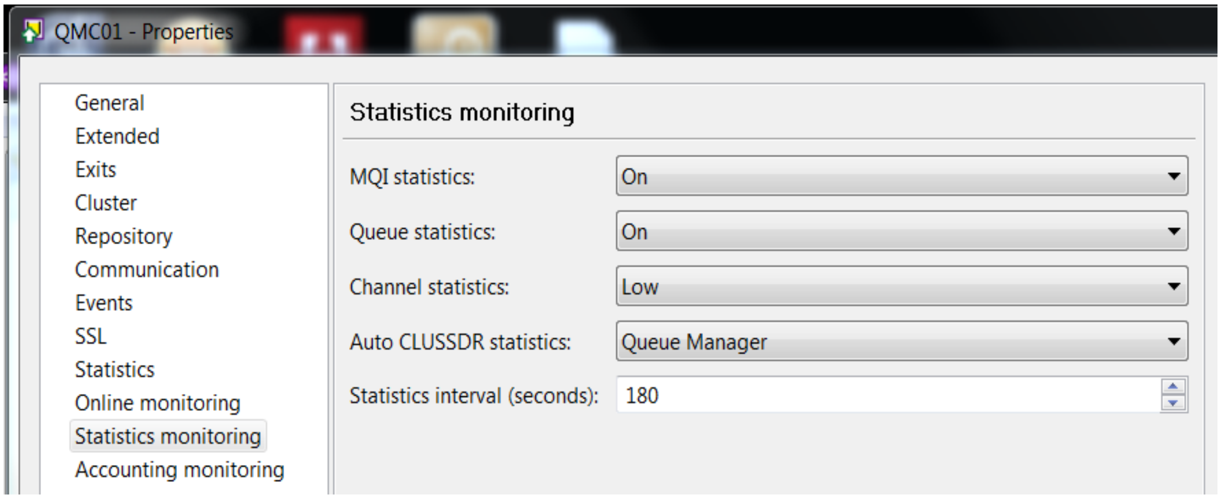
   __ b. To work with statistics properties, click **Statistics monitoring** in the **Properties** list.

   Five queue manager properties control statistics generation:

   - **MQI statistics** (STATMQI) counts the number of MQ API requests processed in the recording interval.
   - **Queue statistics** (STATQ) records information about every local queue in the system. When a queue is accessed indirectly, for example by using a queue alias, the statistics are collected against the resolved local queue. Actions against remote queues are recorded against the associated transmission queue.
   - **Channel statistics** (STATCHL) records information about every channel (except SVRCONNs) in the system.
   - **Auto CLUSSDR statistics** (STATACLS) specifies whether to collect statistics data about the activity of auto-defined cluster-sender channels.
   - **Statistics interval** (STATINT) specifies, in seconds, the recording interval for recording data.

__ c. Activate statistics monitoring. In the **Statistics monitoring** window, set the following properties:

- MQI statistics: **On**
- Queue statistics: **On**
- Channel statistics: **Low**
- Statistics interval: **180**



Statistics monitoring

MQI statistics: On
Queue statistics: On
Channel statistics: Low
Auto CLUSSDR statistics: Queue Manager
Statistics interval (seconds): 180

With a **Statistics interval** of 180, statistics messages are generated every 3 minutes. The interval is set to a short interval to demonstrate this function. On production queue managers, the interval is likely to be set to a greater interval.

__ d. Click **Apply** and then click **OK**.

These settings enable statistics collection on the queue manager.

Note

You can disable statistics generation on each queue or channel by setting the **Queue statistics** (STATQ) or **Channel statistics** (STATCHL) attribute of the object to **Off**.

Part 2: *Generate and view statistics data*

In this step, you use the IBM MQ sample programs to:

- Put and get messages from a local queue
- Generate local queue statistics to put messages on a remote queue
- Generate statistics data for a channel

After the statistics are generated, you review the statistics data.

Note

This part of the exercises assumes that you successfully completed Exercise 4, “Connecting queue managers”.

__ 1. If it is not running, start the queue manager QMR01 and verify that the listener is running.

__ 2. On QMR01, define a local queue that is named **STATS.Q1**.

__ 3. On QM01, define a local queue that is named **STATS.Q1**.

__ 4. On QM01, define a local queue that is called **STATS.Q2**.

__ 5. On QM01, define a remote queue that is called **STATS.Q1.REMOTE** that points to the STATS.Q1 queue on queue manager QMR01.

Set the following remote queue definition properties:

- Remote queue: **STATS.Q1**
- Remote queue manager: **QMR01**
- Transmission queue: **QMR01**

__ 6. On QM01, verify that you have three queues on QM01 for statistics:

- A local queue that is named **STATS.Q1**
- A local queue that is named **STATS.Q2**
- A remote queue definition that is named **STATS.Q1.REMOTE**

__ 7. On QM01, start the sender channel QM01.QMR01.

Ensure that the channel is in the “Running” status before you continue.

Note

The QM01.QMR01 channel is the same channel that was used in Exercise 4, “Connecting queue managers”.

__ 8. Run the **`amqsput`** sample program to put messages to the remote queue definition on queue manager QM01 to send messages over the channel from QM01 to QMR01. Input messages to the queue by using **`statsdata.txt`** file, which contains 536 messages.

This command writes messages to the transmission queue (QMR01), which then is sent across the channel to queue manager QMR01.

Windows

Type: **`amqsput STATS.Q1.REMOTE QM01 < C:\labfiles\Lab12\data\statsdata.txt`**

Linux

Type:
`amqsput STATS.Q1.REMOTE QM01 < /home/localuser/labfiles/Lab12/data/statsdata.txt`

__ 9. Verify that you have 536 messages on STATS.Q1 on QMR01.

__ 10. Use the **amqsput** program to write messages to STATS.Q2 on QM01.

Windows

Type: **`amqsput STATS.Q2 QM01 < C:\labfiles\Lab12\data\statsdata.txt`**

Linux

Type: **`amqsput STATS.Q2 QM01 < /home/localuser/labfiles/Lab12/data/statsdata.txt`**

__ 11. Verify that you have 536 messages on STATS.Q2 on QM01.

__ 12. Use the **amqsget** sample program to read the messages off the STATS.Q2 queue.

In the command window, type:

```
amqsget STATS.Q2 QM01
```

__ 13. After the **amqsget** sample program ends, use the MQ sample program **amqsmon** to read the statistics messages from SYSTEM.ADMIN.STATISTICS.QUEUE on QM01.

Return to the window that you used to run **amqsmon**.

Windows

Type: **`amqsmon -m QM01 -t statistics > c:\labfiles\Lab12\statistics.txt`**

Linux

Type:
`amqsmon -m QM01 -t statistics > /home/localuser/labfiles/Lab12/data/statistics.txt`

__ 14. Open file **`statistics.txt`** in a text editor.

__ 15. The file contains several statistics messages that your tests generate. There are at least three statistics messages that are generated in each statistics interval (that is, as it is specified on the **Statistics Interval Queue Manager** property).

One message shows queue manager statistics. One message is generated for each queue to show queue statistics. One message is generated for the channel to show channel statistics.

Examine the file and locate the following messages:

- The queue manager statistics message for the test period
- The queue statistics message for STATS.Q2 for the test period
- The queue statistics message for QMR01 (that is, the transmission queue that is used to send messages to your other queue manager)
- The channel statistics message for QM01.QMR01 for the test period

__ 16. Disable statistics monitoring for QM01.

__ a. In the **MQ Explorer - Navigator** view, right-click **QM01** and then click **Properties**.

__ b. Click **Statistics monitoring** in the **Properties** list.

__ c. To deactivate statistics monitoring, set the following properties in **Statistics monitoring** window:

○ MQI statistics: **Off**
○ Queue statistics: **Off**
○ Channel statistics: **Off**

__ d. Click **Apply** and then click **OK**.

Part 3: Configure a queue manager to collect accounting data

In this part of the exercise, you configure a queue manager to collect accounting data. The configuration is controlled by using a set of queue manager, queue, and channel properties. These properties can be set by using MQ Explorer or MQSC commands; the instructions in this exercise are for MQ Explorer.

__ 1. In the **MQ Explorer - Navigator** view, right-click **QM01** and then click **Properties**. The properties for the QM01 queue manager are opened.

__ 2. To work with accounting properties, click **Accounting monitoring** in the **Properties** list.

Four queue manager properties control accounting data generation:

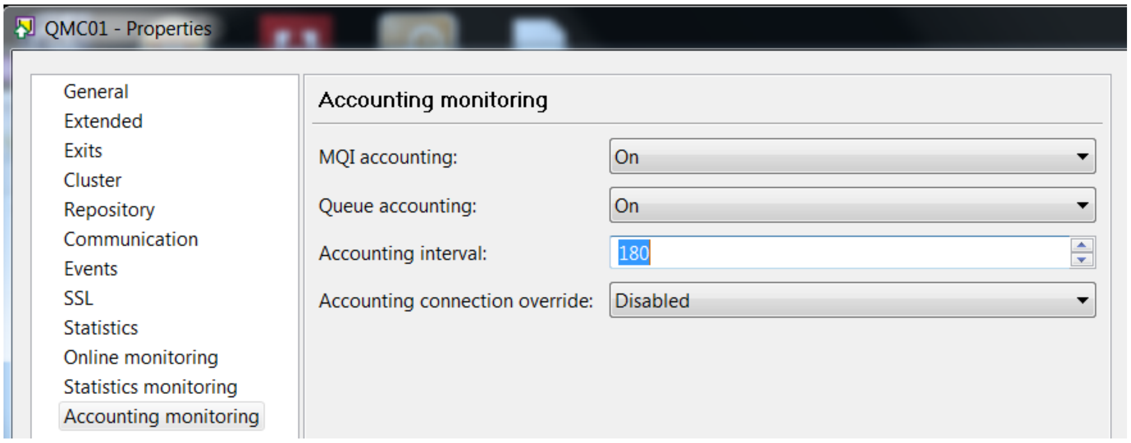
- **MQI accounting** (ACCTMQI) records collected information about the MQI operations that each application runs.
- **Queue accounting** (ACCTQ) records collected information about queues. Messages contain lists of records, one for each local queue open during the time interval.
- **Accounting interval** (ACCTINT) defines the interval in which to write intermediate records (that is, for long-running applications). Setting this attribute to zero turns off intermediate records.

- **Accounting connection override** (ACCTCONO) enables applications to override the attributes by using the **Connect** options in MQCONNX calls.

__ 3. Activate accounting data recording on QM1. In the **Accounting monitoring** properties window, set the following attributes:

- MQI accounting: **On**
- Queue accounting: **On**
- Accounting interval: **180**
- Accounting connection override: **Disabled**

__ 4. Click **Apply** and then click **OK**.


QMC01 - Properties

General
Extended
Exits
Cluster
Repository
Communication
Events
SSL
Statistics
Online monitoring
Statistics monitoring
Accounting monitoring

Accounting monitoring

MQI accounting: On
Queue accounting: On
Accounting interval: 180
Accounting connection override: Disabled

Part 4: Generate and view accounting data

In this part of the exercise, you run IBM MQ sample programs to simulate MQ activity and use the sample program to read and format the generated PCF messages. The test programs write messages to a local queue, and then read them back off the queue.

At the end of this part of the exercise, you review the generated accounting data.

__ 1. Create a local queue on QM01 that is named **ACCTG.STATS.Q1**.

__ 2. From a command window, run the **`amqsput`** program with the **`statsdata.txt`** file to put messages to the local queue, ACCTG.STATS.Q1. The **`statsdata.txt`** file is a text file of 536 lines, which generates a message for each line.

Windows

Type: **`amqsput ACCTG.STATS.Q1 QM01 < C:\labfiles\Lab12\data\statsdata.txt`**

Linux

Type:

```
amqsput ACCTG.STATS.Q1 QM01 </home/localuser/labfiles/Lab12/data/statsdata.txt
```

__ 3. Use MQ Explorer to check the queue depth of queue ACCTG.STATS.Q1 and verify that the queue contains 536 messages.

__ 4. Use the **amqsget** program to read messages from ACCTG.STATS.Q1.

In the command window, type:

```
amqsget ACCTG.STATS.Q1 QM01
```

This command gets all the messages off the queue.

__ 5. Check the queue depth of ACCTG.STATS.Q1. It should now be 0.

__ 6. Display the accounting data that is generated as a result of using **amqsput** and **amqsget**.

MQ provides a sample program that can be used to read and format statistics and accounting data.

Use the MQ sample program **amqsmon** to read the accounting PCF messages from SYSTEM.ADMIN.ACCOUNTING.QUEUE and write the results to text file.

Windows

Type: **amqsmon -m QM01 -t accounting > c:\labfiles\Lab12\accounting.txt**

Linux

Type:

```
amqsmon -m QM01 -t accounting > /home/localuser/labfiles/Lab12/accounting.txt
```

__ 7. Open the file **accounting.txt** in a text editor.

The file contains several messages that the **amqsput** and **amqsget** sample programs generate when you use them.

Look at the file and locate the messages that are associated with **amqsput** and **amqsget**. Each program (**amqsput** and **amqsget**) generates two messages. One message is generated to show MQI accounting, that is, the MQI operations that get each application runs. The other message is generated to show queue accounting, that is, the details of the queue that the application uses.

Note

For applications that use multiple queues, a separate message exists for each queue.

Look at the file and locate the MQI accounting and queue accounting messages for each application.

Table 1 and Table 2 describe the fields in each message.

Table 1. MQI accounting fields

Field	Description
QueueManager	Name of the queue manager.
IntervalStartDate	Date of the start of the monitoring period.
IntervalStartTime	Time of the start of the monitoring period.
IntervalEndDate	Date of the end of the monitoring period.
IntervalEndTime	Time of the end of the monitoring period.
CommandLevel	IBM MQ command level.
ConnectionId	Connection ID for the task.
SeqNumber	Sequence number (normally zero). Increments for each subsequent record for long running connections.
ApplicationName	Name of the application (program name).
ApplicationPid	Operating system process ID of the application.
ApplicationTid	IBM MQ thread ID of the connection in the application.
User ID	User ID context of the application.
ConnDate	Normally matches “StartTime,” except for long-running connections.
ConnTime	Normally matches “StartTime,” except for long-running connection.
ConnName	Only written for client connections.
ChannelName	Only written for client connections.
DiscDate	Normally matches “EndTime,” except for long running connections.
DiscTime	Normally matches “EndTime,” except for long running connections.
DiscType	MQ_DISCONNECT_TYPE_NORMAL, application requested. MQ_DISCONNECT_TYPE_IMPLICIT, abnormal application termination. MQ_DISCONNECT_TYPE_QMGR, queue manager broke the connection (`shutdown -i`).
OpenCount	Number of objects opened.
OpenFailCount	Number of unsuccessful open object attempts.
CloseCount	Number of objects closed.
CloseFailCount	Number of unsuccessful close object attempts.
PutCount	Number of messages that are successfully put to a queue (excluding MQPUT1 requests).
PutFailCount	Number of unsuccessful put message attempts.
PutBytes	Number bytes written in MQPUTs.
GetCount	Number of successful destructive MQGETs.

Table 1. MQI accounting fields

Field	Description
GetFailCount	Number of unsuccessful destructive gets.
GetBytes	Total number of bytes retrieved.
BrowseCount	Number of successful non-destructive gets.
BrowseFailCount	Number of successful non-destructive gets.
BrowseBytes	Total number of bytes browsed.
CommitCount	Number of successful transactions that are completed including those transactions that the application commits when it disconnects.
CommitFailCount	Number of unsuccessful attempts to complete a transaction that include those transactions that run an implicit commit because the application disconnected.
BackCount	Number of backouts processed (including implicit backout upon abnormal disconnect).
InqCount	Number of successful objects inquired upon.
InqFailCount	Number of unsuccessful “object inquire” attempts.
SetCount	Number of successful object updates (SET).
SetFailCount	Number of successful SET attempts.

Note: The OpenCount, OpenFailCount, CloseCount, and CloseFailCount fields are displayed as arrays.

Table 2. Queue accounting fields

QueueManager	Name of the queue manager
IntervalStartDate	Date of the start of the monitoring period.
IntervalStartTime	Time of the start of the monitoring period.
IntervalEndDate	Date of the end of the monitoring period.
IntervalEndTime	Time of the end of the monitoring period.
CommandLevel	IBM MQ command level.
ConnectionId	Connection ID for the task.
SeqNumber	Sequence number normally zero. Increments for each subsequent record for long running connections.
ApplicationName	Name of the application (program name).
ApplicationPid	Operating system process ID of the application.
ApplicationTid	IBM MQ thread ID of the connection in the application.
User ID	User ID context of the application.
ObjectCount	Number of queues that are accessed in the interval for which accounting data were recorded.
QueueName	Name of the queue.

Table 2. Queue accounting fields

QueueManager	Name of the queue manager
CreateDate	Date the queue was created.
CreateTime	Time the queue was created.
QueueType	Type of the queue.
QueueDefinitionType	Queue definition type. Valid values are: • PREDEFINED • PERMANENT_DYNAMIC • TEMPORARY_DYNAMIC
OpenCount	Number of times the application opened this queue in this interval.
OpenDate	Date of the first open of the queue in this recording interval. If the queue was already open at the start of this interval, this value is the date that the queue was originally opened.
OpenTime	Time of first open of the queue in this recording interval. If the queue was already open at the start of this interval, this value is the time that the queue was originally opened.
CloseCount	Number of objects closed.
CloseDate	Date of final close of the queue in this recording interval. If the queue is still open, then the value is not returned.
CloseTime	Time of final close of the queue in this recording interval. If the queue is still open, then the value is not returned.
PutCount	Number of messages that are successfully put to the queue (excluding MQPUT1 requests).
PutFailCount	Number of unsuccessfully put message attempts.
Put1Count	Number of messages that are successfully put to the queue through MQPUT1 requests.
Put1FailCount	Number of unsuccessful MQPUT1 message attempts.
PutBytes	Total number of bytes put.
PutMinBytes	Smallest message size that is placed on the queue.
PutMaxBytes	Largest message size that is placed on the queue.
GetCount	Number of successful destructive MQGETs.
GetFailCount	Number of unsuccessful destructive GETs.
GetBytes	Number of bytes read in destructive GETs.
GetMinBytes	Size of the smallest messages that are retrieved from the queue.
GetMaxBytes	Size of the largest messages that are retrieved from the queue.
BrowseCount	Number of successful non-destructive GETs.
BrowseFailCount	Number of unsuccessful non-destructive MQGETs.

Table 2. Queue accounting fields

QueueManager	Name of the queue manager
BrowseBytes	Number of bytes read in non-destructive MQGETs that returned persistent messages.
BrowseMinBytes	Size of the smallest messages that are browsed from the queue.
BrowseMaxBytes	Size of the largest messages that are browsed from the queue.
GeneratedMsgCount	Number of messages generated (report / event).
TimeOnQMin	Shortest time a message remained on the queue before it is retrieved (in microseconds).
TimeOnQAvg	Average time a message remained on the queue before it is retrieved (in microseconds).
TimeOnQMax	Longest time a message remained on the queue before it is retrieved (in microseconds).

__ 8. Disable accounting monitoring on the QM01 queue manager.

- __ a. In the **MQ Explorer - Navigator** view, right-click **QM01** and then click **Properties**. The properties for the QM01 queue manager are opened.
- __ b. Click **Accounting monitoring** in the **Properties** list.
- __ c. Deactivate accounting data recording on QM1. In the **Accounting monitoring** properties window, set the following attributes:
  - ○ MQI accounting: **Off**
  - ○ Queue accounting: **Off**
- __ d. Click **Apply** and then click **OK**.

Part 5: Configure a queue manager for online monitoring

In this section, you configure a queue manager for online monitoring. By default, online monitoring data is not collected. It must be enabled by using a set of queue manager attributes.

The configuration of monitoring data collection has three values to enable the collection of data (LOW, MEDIUM, and HIGH). Collecting monitoring data might require that you run some instructions that affect performance (for example, obtaining system time).To reduce the impact of online monitoring, the resource-intensive measurements are sampled instead of measuring every value.

The online monitoring values can be set on each queue manager. The configuration is controlled by using a set of queue manager, queue, and channel attributes. These values can be set by using the MQ Explorer or MQSC commands; the instructions in this exercise are for MQ Explorer.

__ 1. In MQ Explorer, open the properties for queue manager QM01.

__ 2. Select **Online monitoring**.

Three queue manager properties control online monitoring:

- **Channel monitoring** (MONCHL) controls the collection of online monitoring data for channels.
- **Queue monitoring** (MONQ) controls the collection of online monitoring data for queues.
- **Auto CLUSSDR monitoring** (MONACLS) controls the collection of monitoring data for cluster sender channels.

__ 3. Activate recording of statistics for channels and queues. In the **Online Monitoring** properties for QM01, set the following values:

- Channel monitoring: **High**
- Queue monitoring: **High**

__ 4. Run the **`amqsput`** sample program with the **`statsdata.txt`** file to put 536 messages on STATS.Q2.

__ 5. Run the **`amqsget`** sample program against STATS.Q2 to get the messages.

__ 6. Review the generated online monitoring data for QM01 by using MQSC commands.

In the MQSC session for QM01, type:

```
DIS QSTATUS(STATS.Q2) MONITOR
```

This command displays the monitoring data for the queue. Table 3 summarizes the monitoring fields.

Table 3.

QTIME	The interval between when messages are put on the queue and when they are destructively read. The value is shown in microseconds, formatted into a string of up to 9 digits.
LPUTDATE	Date on which the last message was put to the queue since the queue manager started. If no PUT date is available, then the value is blank.
LPUTTIME	Time at which the last message was put to the queue since the queue manager start. If no put time is available, the value shows as blank.
LGETDATE	Date on which the last message was retrieved from the queue since the queue manager start. A message that is browsed does not count as a retrieved message. If no get date is available, the value is shown as blank.
LGETTIME	Time at which the last message was retrieved from the queue since the queue manager start. A browsed message does not count as a retrieved message. If no get time is available, the value is shown as blank.
MONQ	Current level of monitoring data collection for the queue. Possible values: OFF, LOW, MEDIUM, HIGH.
MSGAGE	Age, in seconds, of the oldest message on the queue. The maximum displayable value is 999 999 999. If the age exceeds this value, 999 999 999 is displayed.

__ 7. Disable online monitoring on the QM01 queue manager.

__ a. In MQ Explorer, open the properties for queue manager QM01.

__ b. Click **Online monitoring**.

__ c. Deactivate recording of statistics for channels and queues. In the **Online Monitoring** properties for QM01, set the following values:

- Channel monitoring: **Off**
- Queue monitoring: **Off**

__ d. Click **Apply** and then click **OK**.

Part 6: Resource monitoring

In this part of the exercise, you use the **amqsrua** sample program to enable and display MQ resource monitoring data. The **amqsrua** command reports metadata that is published by queue managers. This data can include information about the CPU, memory, and disk usage.

In this part of the exercise, you need two command windows (sessions). In one window, you run the **amqsrua** sample program. In the second window, you run the commands to put and get messages to a queue on a QM01

__ 1. Use the **amqsrua** sample program in interactive mode to get a queue status summary.

__ a. In one of the command windows, type:

```
amqsrua -m QM01
```

__ b. The command responds with a set of resource classes: CPU, DISK, STATMQI, and STATQ.

Type `STATQ` and then press Enter.

__ c. You can now specify the type of action you want to monitor for a queue: OPENCLOSE, INQSET, PUT, or GET.

Type `PUT` and then press Enter.

__ d. Specify the name of the queue to monitor.

Type `STATS.Q1` and then press Enter.

The **amqsrua** sample program is now subscribing to the specified class of resource monitoring events. Using the command in this way reports on the values of those resource elements every 10 seconds. Currently, the queue is empty so all values are zero.

__ 2. In the second command window, run the **amqsput** program to put messages to the STATS.Q1 queue on QM01.

Windows

Type: **`amqsput STATS.Q1 QM01 < C:\labfiles\Lab12\data\statsdata.txt`**

Linux

Type: **`amqsput STATS.Q1 QM01 < /home/localuser/labfiles/Lab12/data/statsdata.txt`**

__ 3. Observe the counts in queue status window. You should see that the MQPUT non-persistent message count is now 536.

__ 4. End the **amqsrua** program. Type: Ctrl+C

__ 5. Run the **amqsrua** program in interactive mode to get the log statistics.

__ a. In the command windows, type:

```
amqsrua -m QM01
```

__ b. For the class selection, type **DISK** and then press Enter.

__ c. For the type selection, type **Log** and then press Enter.

The Log statistics include the bytes in use, the maximum bytes available, physical bytes written, logical bytes written, and write latency.

The command should return output that is similar to the following example:

```
Publication received PutDate: 20161027 PutTime: 19064332 Interval: 10.00
seconds
Log - bytes in use 50331648
Log - bytes max 83886080
Log - physical bytes written 4096 19/sec
Log - logical bytes written 1352 6/sec
Log - write latency 15539 uSec
```

__ d. End the **amqsrua** program.

__ 6. Run the **amqsrua** program in command-line mode to get message data on directories for errors, trace, and the number of MQ FDC records. Type:

```
amqsrua -m QM01 -c DISK -t SystemSummary
```

The command should return output similar to the following example:

```
Publication received PutDate: 20161027 PutTime: 19064332 Interval: 10.00
seconds
MQ errors file system - bytes in use 204521MB
MQ errors file system - free space 57.11%
MQ FDC file count 4
MQ trace file system - bytes in use 204521MB
MQ trace file system - free space 57.11%
```

__ 7. The **`amqsrua`** sample program subscribes to the MQ resource monitoring topics. Verify the subscription and the topic string in MQ Explorer.

__ a. Click **Subscriptions** under QM01 in the **MQ Explorer - Navigator** view.

__ b. You should see one subscription for the topic string: **`$SYS/MQ/INFO/QMGR/QM01/Monitor/DISK/SystemSummary`**

Information

You can run the **`amqsrua`** program with the argument **`-d 1`** to enable debug mode. The results show the system topics that the program subscribes to.

__ 8. End the **`amqsrua`** program.

End of exercise

Exercise review and wrap-up

You should now be able to:

- Use the IBM MQ Explorer to enable accounting, statistics, and online monitoring collection
- View accounting, statistics, and online monitoring data
- Use the **`amqsrua`** sample program to subscribe to MQ resource monitoring topics

IBM Training