



IBM Washington
Systems
Center

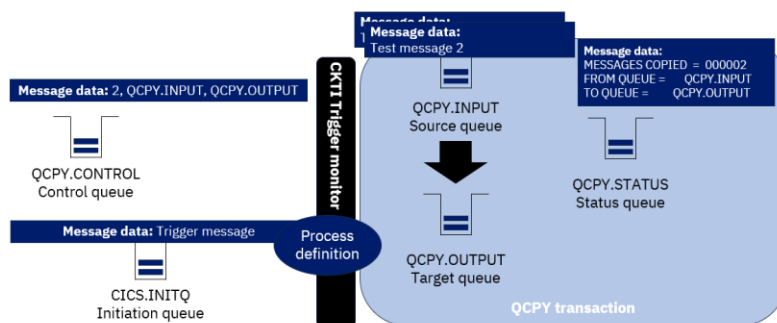
How to use triggering with IBM MQ for z/OS

Introduction:

This lab walks you through a triggering example using a sample COBOL MQ CICS program, QCPY. The QCPY program is executed from the QCPY transaction. It copies messages from one queue to another applying a message property to each message. It is started by a comma delimited control message which triggers the transaction. It also uses information from the IBM MQ process object. sample requires a currently supported version of IBM MQ and CICS. You can find the source code in COBOL for this application in ZQS1.COBOL.SOURCE.

QCPY Program Flow

1. The QCPY transaction is triggered.
2. The control queue QCPY.CONTROL is opened.
3. The copy control message is read.
4. Control message is parsed into the controlling fields.
5. The source queue QCPY.INPUT is opened.
6. The target queue QCPY.OUTPUT is opened.
7. In a loop, messages are read from the source queue and written to the target queue.
8. The status message is built.
9. The status queue QCPY.STATUS opened and the status message is put.
10. All queues are closed.
11. Control is returned to CICS.



Lab Instructions:

1. First thing we need to do is ensure CICS is running. We can test this via sdsf from the main menu

```
Menu Utilities Compilers Options Status Help
ISPF Primary Option Menu Invalid option

0 Settings      Terminal and user parameters   User ID . : DQUINCY
1 View          Display source data or listings  Time. . . : 14:14
2 Edit          Create or change source data    Terminal. : 3278
3 Utilities     Perform utility functions      Screen. . : 1
4 Foreground    Interactive language processing  Language. : ENGLISH
5 Batch         Submit job for language processing Appl ID . : ISR
6 Command       Enter TSO commands              TSO logon : IKJACCT
7 Dialog Test   Perform dialog testing          TSO prefix: DQUINCY
9 IBM Products  IBM program development products System ID : MQS1
10 SCLM         SW Configuration Library Manager MVS acct. : ACCNT#
11 Workplace    ISPF Object/Action Workplace     Release . : ISPF 8.1
12 z/OS System  z/OS system programmer applications
13 z/OS User    z/OS user applications

Enter X to Terminate using log/list defaults

Option ==> SDSF_
```

2. Navigate to 'da' once in the SDSF menu to see active users.

- Set the prefix to * so we can see all active users with the command 'prefix *'. Then, using the F7 and F8 keys, navigate to see if CICS is running. You should see something

Display	Filter	View	Print	Options	Search	Help						
SDSF	DA	MQS1	MQS1	PAG	0	CPU	2	LINE 51-67 (67)				
NP	JOBNAME	StepName	ProcStep	JobID	Owner	C	Pos	DP	Real	Paging	SIO	
	AXR04	AXR04		STC08891	SYSPROG	L0	FF	610	0.00	0.00		
	FTP1	FTP1		STC08892	SYSPROG	L0	FF	675	0.00	0.00		
	INETD1	INETD1		STC08893	SYSPROG	L0	FF	475	0.00	0.00		
	SYSLOGD	SYSLOGD		STC08894	SYSPROG	NS	C1	728	0.00	0.00		
	ZQS1MSTR	ZQS1MSTR	PROCSTEP	STC08915	SYSPROG	NS	FE	33T	0.00	0.00		
	GPMSERVE	RMFDDS01	STEP1	STC08916	SYSPROG	NS	FE	6890	0.00	0.00		
	TCPIP	TCPIP	TCPIP	STC08898	SYSPROG	NS	FE	8343	0.00	0.00		
	RMFGAT	RMFGAT	IEFPROC	STC08899	SYSPROG	NS	FE	21T	0.00	0.00		
	D3A1ADMT	D3A1ADMT	STARTADM	STC08913	SYSPROG	IN	FE	2764	0.00	0.00		
	D3A1DBM1	D3A1DBM1	IEFPROC	STC08911	SYSPROG	NS	FE	66T	0.00	0.00		
	D3A1IRLM	D3A1IRLM		STC08910	SYSPROG	NS	FE	10T	0.00	0.00		
	TN3270	TN3270	TN3270	STC08904	SYSPROG	NS	FE	2465	0.00	0.00		
	D3A1DIST	D3A1DIST	IEFPROC	STC08912	SYSPROG	NS	FE	4791	0.00	0.00		
	MQS1CICS	MQS1CICS	CICS	STC08914	CICSSTC	NS	FE	25T	0.00	0.00		
	BP01	BP01	BP01			NS	FE	1336	0.00	0.00		
	D3A1MSTR	D3A1MSTR	IEFPROC	STC08909	SYSPROG	NS	FE	2487	0.00	0.42		
	PORTMAP	PORTMAP	PORTMAP	STC08902	SYSPROG	L0	FF	515	0.00	0.00		
COMMAND INPUT ==>											SCROLL ==> CSR	

like this:

- If there is no CICS region active, you will need to start the cics1 region w/ command 'start cics1'
- To navigate to CICS, start another MQS1 PCOMM session and use the MQS1CICS command

```
MSG10
WSC MQPLEX1

Enter: MQS1CICS_
MQSx      - TSO on MQSx (MQS1 or MQS2)
MQSxCICS  - for CICS on MQSx (MQS1CICS or MQS2CICS)
```

WELCOME TO CICS 14:20:01

```
*****\ *****\ *****\ *****\ (R)
*****\ *****\ *****\ *****\
**\\\\**\ **\\ **\\\\**\ **\\\\**\
**\  \\ **\ **\  \\ **\  \\
**\      **\ **\      *****\
**\      **\ **\      *****\
**\      **\ **\      \\\**\
**\  **\ **\  **\  **\  **\  **\
*****\ *****\ *****\ *****\
*****\\ *****\\ *****\\ *****\\
\\\\\\  \\\  \\\  \\\  \\\  \\\
```

6. From the CICS main screen, hit tab once then type in CKQC. This is the MQ CICS transaction CKQC. This transaction makes it possible to monitor and control the interface between MQ and CICS.
- 7.

```
----- Connection -----
CKQCM0          CICS Adapter Control -- Initial panel
Select menu bar item using Tab key. Then press Enter.

F1=Help  F3=Exit
```

8. Now navigate to MQ Explorer.
9. You will need to define several queue objects:
 - a. QCPY.CONTROL
 - b. QCPY.INPUT
 - c. QCPY.OUTPUT
 - d. QCPY.STATUS

10. You can see the properties for all of these queues below

QCPY.CONTROL

Control Message – the message used to start the QCPY transaction. For QCPY the message contains, in comma delimited format:

1. The number of messages to be copied
2. The source queue
3. The target queue

The screenshot shows the 'QCPY.CONTROL - Properties' dialog box with the 'Triggering' tab selected. The left sidebar lists 'General', 'Extended', 'Cluster', 'Triggering' (highlighted), 'Events', 'Storage', and 'Statistics'. The main area contains the following fields:

- Trigger control: On (dropdown)
- Trigger type: Every (dropdown)
- Trigger depth: 1 (spinner)
- Trigger message priority: 0 (spinner)
- Trigger data: I (text field)
- Initiation queue: CICS01.INITQ (text field with 'Select...' button)
- Process name: QCPY.PROCESS (text field)

QCPY.INPUT - The source of the messages to be copied.

The screenshot shows the 'QCPY.INPUT - Properties' dialog box with the 'Triggering' tab selected. The left sidebar lists 'General', 'Extended', 'Cluster', 'Triggering' (highlighted), 'Events', 'Storage', and 'Statistics'. The main area contains the following fields:

- Trigger control: Off (dropdown)
- Trigger type: First (dropdown)
- Trigger depth: 1 (spinner)
- Trigger message priority: 0 (spinner)
- Trigger data: (empty text field)
- Initiation queue: (empty text field with 'Select...' button)
- Process name: (empty text field)

QCPY.OUTPUT – The target for the copied messages.

QCPY.OUTPUT - Properties

General

Extended

Cluster

Triggering

Events

Storage

Statistics

Triggering

Trigger control: Off

Trigger type: First

Trigger depth: 1

Trigger message priority: 0

Trigger data:

Initiation queue: Select...

Process name:

QCPY.STATUS – The queue which will hold the status messages, reporting on success or failure

QCPY.STATUS - Properties

General

Extended

Cluster

Triggering

Events

Storage

Statistics

Triggering

Trigger control: Off

Trigger type: First

Trigger depth: 1

Trigger message priority: 0

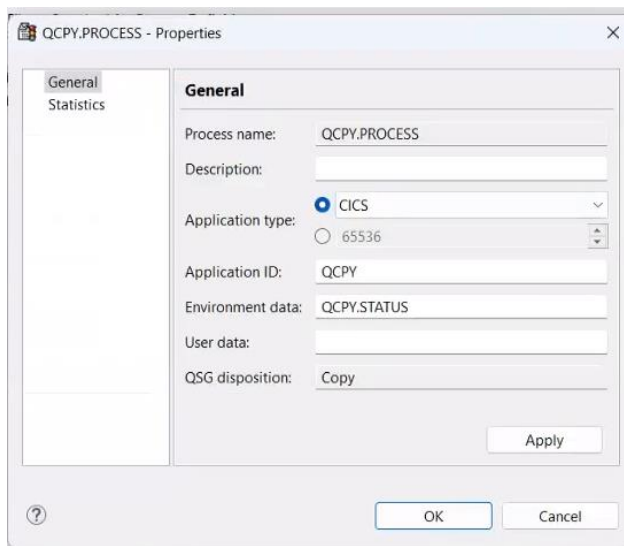
Trigger data:

Initiation queue: Select...

Process name:

11. You will also need a process definition. A process is an MQ object that defines an application to the MQ Queue Manager. The process definition is used to identify applications to be started by a trigger monitor. It includes application ID and type, plus some application specific data.
- Here, we'll specify CICS as our application and give it the ID 'QCPY'.
 - Application ID which is the transaction name in CICS
 - Environment data is status queue which tells us what happen at the end of the process

QCPY.PROCESS



The screenshot shows the 'QCPY.PROCESS - Properties' dialog box with the 'General' tab selected. The dialog has a left sidebar with 'General' and 'Statistics' tabs. The main area contains the following fields:

- Process name: QCPY.PROCESS
- Description: (empty text box)
- Application type: ☒ CICS (selected), ☐ 65536 (unselected)
- Application ID: QCPY
- Environment data: QCPY.STATUS
- User data: (empty text box)
- QSG disposition: Copy

At the bottom right is an 'Apply' button. At the bottom center are 'OK' and 'Cancel' buttons. A help icon (?) is at the bottom left.

12. Now that we have our queues configured, we will use the QCPYLOAD JCL is used to load up messages onto our queue. QCPYLOAD puts test messages onto QCPY.INPUT using OEMPUT program.

Message browser

Queue Manager Name: ZMQ1

Queue Name: QCPY.INPUT

✓ Position	Put date/time	User identifier	Put application name	Format	Total length
1	Mar 15, 2024 1:58:54 PM	LYNELKINS	MQ Explorer 9.3.0	MQSTR	5
2	Mar 15, 2024 1:58:58 PM	LYNELKINS	MQ Explorer 9.3.0	MQSTR	5
3	Mar 15, 2024 1:59:04 PM	LYNELKINS	MQ Explorer 9.3.0	MQSTR	4
4	Mar 15, 2024 1:59:08 PM	LYNELKINS	MQ Explorer 9.3.0	MQSTR	4
5	Mar 15, 2024 1:59:12 PM	LYNELKINS	MQ Explorer 9.3.0	MQSTR	5

Copies from QCPY.INPUT to QCPY.OUTPUT based on a control card that we give it in QCPY.CONTROL

```

File Edit Edit Settings Menu Utilities Compilers Test Help
EDIT      ELKINSC.COBOL.JCL(QCYPST1) - 01.00      Columns 00001 00072
Command ==> _      Scroll ==> CSR
***** ***** Top of Data *****
==MSG> -CAUTION- Profile changed to CAPS ON (from CAPS OFF) because the
==MSG> data does not contain any lower case characters.
==MSG> -Warning- The UNDO command is not available until you change
==MSG> your edit profile using the command RECOVERY ON.
000001 5,QCOPYPRP.INPUT.QUEUE,QCOPYPRP.OUTPUT.QUEUE,TEST_PROP
***** ***** Bottom of Data *****

```

The input here is: number of messages you want to copy, the input queue, the output queue

13. Currently, no one is listening, so we'll need to add a listener to CICS1.INITQ. Set up a CICS.INITQ local queue with the following properties:

CICS01.INITQ - Properties

General

Extended

Cluster

Triggering

Events

Storage

Statistics

General

Queue name: CICS01.INITQ

Queue type: Local

QSG disposition: Queue manager

Description: CKTI initiation queue

Put messages: Allowed

Get messages: Allowed

Default priority: 5

Default persistence: Persistent

Usage: Normal

14. From z/OS CICS screen, navigate to CKQCM0 by typing in the command:

```
WELCOME TO CICS 16:25:11

*****\ *****\ *****\ *****\ (R)
*****\ *****\ *****\ *****\
**\ |||**\ **\ |||**\ **\ |||**\
**\  ||  **\  **\  ||  **\  ||
**\      **\  **\      *****\
**\      **\  **\      *****\
**\      **\  **\      |||**\
**\  **\  **\  **\  **\  **\  **\
*****\ *****\ *****\ *****\
*****\ *****\ *****\ *****\
||| ||| ||| ||| ||| ||| ||| |||

DFHAC2001 04/17/2024 16:25:14 MQS1CICS Transaction '' is not recognized. Check
that the transaction name is correct. CKQCM0_
```

15. This screen should pop up.

```

Connection      CKTI      Task
-----
CKQCM0          CICS Adapter Control -- Initial panel

Select menu bar item using Tab key. Then press Enter.


F1=Help  F3=Exit
```

16. Hit the tab button. The following menu will pop up. Type in option 1 and hit enter.

```

Connection      CKTI      Task
+-----+
| Select an action. | apter Control -- Initial panel
|                  |
| 1. Start...      | sing Tab key. Then press Enter.
| 2. Stop...       |
| 3. Modify...     |
| 4. Display       |
|                  |
+-----+
| F1=Help F12=Cancel |
+-----+
```

17. Enter in the following details and hit enter:

Connection	CKTI	Task
CKQCM0	CIC	Select an action.
Select menu bar it	1 1. Start... 2. Stop... 3. Display	initial panel press Enter.
<div> <div>F1</div> <div>Start Task Initiator</div> <div>Type Initiation Queue Name. Then press Enter.</div> <div>Initiation Queue Name (IQ)</div> <div>CICS01.INITQ</div> <div>F1=Help F12=Cancel</div> </div>		

This step initiates the CKTI transaction, which is what controls the CICS trigger monitor.

```

_CKQCM1                CICS adapter messages panel

Read messages. Then press F4 to retry if appropriate, or F12 to cancel.

DFHMQ0386 I MPZ1CIC1 STARTCKTI initiated from TERMID=0031 TRANID=CKSQ
USERID=CICSMQ3 and is accepted.
  
```

18. Put a test message on QCPY.CONTROL to see everything (works).

Put test message

Put message to:

Queue manager:

ZQS1

Queue:

QCPY.CONTROL

Message data:

1, QCPY.INPUT, QCPY.OUTPUT

Put message

Close

Commented [LE1]: This message would only cause 1 message to be copied, not 2

19. This message requests that MQ copies 2 messages from QCPY.INPUT to QCPY.OUTPUT. After you submit this, check it worked by looking at the queue depth.

QCPY.INPUT	Local	Queue man...	0	0	1
QCPY.OUTPUT	Local	Queue man...	0	0	2

20. Next look at the QCPY.STATUS messages. You should see a new message on the queue confirming the QCPY was successful:

```
MESSAGES COPIED = 000002
FROM QUEUE = QCPY.INPUT
TO QUEUE = QCPY.OUTPUT
```

21. Congratulations! You have successfully used a CICS application for triggering! Next, we'll look at what we'd do in a more realistic scenario, where you have many messages flowing through MQ.
22. Navigate to option 3.4 from your ISPF main menu on MQS1.
23. Search for the following dataset: ZQS1.COBOL.JCL

```
Menu RefList RefMode Utilities Help
Data Set List Utility
blank Display data set list P Print data set list
V Display VTOC information PV Print VTOC information
More: +
Enter one or both of the parameters below:
Dsname Level . . . ZQS1.COBOL.JCL
Volume serial . . .
Data set list options
Initial View
1 1. Volume Enter "/" to select option
2. Space / Confirm Data Set Delete
3. Attrib / Confirm Member Delete
4. Total / Include Additional Qualifiers
/ Display Catalog Name
- Display Total Tracks
- Prefix Dsname Level
When the data set list is displayed, enter either:
Option ==>
F1=Help F2=Split F3=Exit F7=Backward F8=Forward F9=Swap
F10=Actions F12=Cancel
09/038
```

24. Choose to browse the data set by putting a 'B' to the left of the name and hitting enter.

```
Menu Options View Utilities Compilers Help
DSLST - Data Sets Matching ZQS1.COBOL.JCL 0 Members processed
Command - Enter "/" to select action Message Volume
-----
B ZQS1.COBOL.JCL Edited MQ1P00
***** End of Data Set List *****
```

25. Once inside the dataset, you'll see a list of several members. Use the command 'sort changed' or navigate the list using F7 and F8 until you see member 'QCPYLOAD'. Place an 'E' to the left of QCPYLOAD and hit enter like so:

Menu	Functions	Confirm	Utilities	Help
BROWSE				
	Name	ZQS1.COBOL.JCL	Size	Created
	QCPYT2		21	2018/08/28
	JC		1	2024/04/18
e	QCPYLOAD		35	2012/08/09
	DEFQCPY		78	2014/03/06
	DEF3QCPY		26	2018/08/27
	QCPYUTL		66	2024/03/15
	CMPLDCPR		46	2024/03/06
	CMPLDCPY		46	2012/08/09
	AFFINITY		1	2012/08/09
	QCPYT1		21	2018/08/28
	QCPYLD2		20	2018/08/27
	QCPYLD3		19	2018/08/28
	TSTPUT1		74	2012/04/04
	CICSMQ7		36	2012/08/16
	QCPYST1		32	2014/03/06
	CMPLDTST		36	2015/03/30
	MKDFQCPY		26	2014/03/06
Command ==>				Row 0000001 of 0000094
F1=Help	F2=Split	F3=Exit	F5=RTFind	F7=Up
F10=Left	F11=Right	F12=Cancel	F8=Down	F9=Swap

26. Browse through the JCL in QCPYLOAD using F7 and F8. You'll notice that this JCL is an execution of OEMPUT. We're going to load up our QCPY.INPUT queue with 500 messages.
27. Submit the JCL using the command line like so:

File	Edit	Edit Settings	Menu	Utilities	Compilers	Test	Help
EDIT							
	ZQS1.COBOL.JCL(QCPYLOAD)	- 01.13					Columns 00001 00072
000016	// SET TEMPFIL='TEAMXX.ZZ2'						
000017	/*SET P='-p' NONP messages						
000018	// SET P=' ' NONP messages						
000019	// SET MSGS=500						
000020	/*						
000021	// EXEC PGM=OEMPUT,REGION=0M,						
000022	// PARM=('-m&QM. -n&MSG. -x &P. -pm')						
000023	//SYSIN DD *						
000024	-qCPY. INPUT						
000025	-fileDD:MSGIN						
000026	-dJTEST4						
000027	//STEPLIB DD DISP=SHR.DSN=&APPLOAD.						
000028	// DD DISP=SHR.DSN=&MLOAD1.						
000029	// DD DISP=SHR.DSN=&MLOAD2.						
000030	// DD DISP=SHR.DSN=&MLOAD3.						
000031	//MSGIN DD DISP=SHR.DSN=ZQS1.COBOL.JCL(MSGS)						
000032	//SYSPRINT DD SYSOUT=*						
000033	/* SUMMARY NOT USED because of concurrent access						
Command ==>	SUBMIT						Scroll ==> CSR

28. Nice! You should receive a RC=0 upon submitting. You can check that the message loading process worked by navigating to MQ Explorer. On MQ Explorer, you will now see QCPY.INPUT has 500 more messages in the queue.
29. Next, navigate back to your terminal display. From here, we will now execute the QCPYT2 job. From the ZQS1.COBOL.JCL members, place an 'b' next to QCPYT2.
30. Before submitting, take a second to see what is being done here. We specify that we want to copy 10 messages from QCPY.INPUT to QCPY.OUTPUT and we will be using the same OEMPUT execution to do this. However, our target queue for OEMPUT here is QCPY.CONTROL, not QCPY.INPUT.

```

BROWSE      ZQS1.COBOL.JCL(QCPYT2) - 01.05      Line
//  SET Q=QCPY.CONTROL
//  SET L=80
//  SET N=1
//PUT01A EXEC PGM=OEMPUT,REGION=0M,
// PARM=(' -m&M -n&N -q&Q -s&L -sr&L -fileDD:MIN')
//SYSIN  DD *
/*
//STEPLIB  DD DISP=SHR,DSN=ZQS1.MP1B.LOAD
//          DD DISP=SHR,DSN=MQ933CD.SCSQLOAD
//          DD DSN=MQ933CD.SCSQANLE,DISP=SHR
//          DD DSN=MQ933CD.SCSQAUTH,DISP=SHR
//SYSPRINT DD SYSOUT=*
//MIN      DD *
10,QCPY.INPUT,QCPY.OUTPUT
/*
//*COR      DD DISP=SHR,DSN=ZQS1.COBOL.JCL(COR01)
//SUMMARY  DD SYSOUT=*

```

31. Enter submit in the command line below the job and hit enter.
32. Now, assuming the job completed successfully, you should be able to look over at MQ Explorer and see 10 messages moved from QCPY.INPUT to QCPY.OUTPUT.
33. That is the QCPY lab! Here, you practiced triggering using a CICS application using a rudimentary example and a more advanced example.