Sample COBOL CICS WMQ Subscription Transaction

The IBM ATS WebSphere MQ team:

Lyn Elkins – elkinsc@us.ibm.com
Mitch Johnson – mitchj@us.ibm.com
Eugene Kuehlthau - ekuehlth@us.ibm.com
Ed Zeilnhofer - edz@us.ibm.com
Introduction........................................................................................................................................3
Terms .................................................................................................................................................3
QSUB – Sample CICS transaction to subscribe to publications .........................................................4
Description........................................................................................................................................4
Inputs..................................................................................................................................................4
   The Subscription Control message .........................................................................................4
   The Trigger message ..............................................................................................................5
Outputs.............................................................................................................................................6
   The status message ................................................................................................................6
QSUBCBL Program Flow ..................................................................................................................7
CICS Definitions ................................................................................................................................8
   QSUB definition .....................................................................................................................8
   QSUBCBL Definition .............................................................................................................10
   QML0GRP Installation .........................................................................................................10
WMQ Definitions ............................................................................................................................12
   Process Definition, QSUB.PROCESS ................................................................................12
   Queue Definitions ..............................................................................................................13
   Topic Object Definition .....................................................................................................18
Testing Samples ..............................................................................................................................19
Testing – Batch Test Submission ...............................................................................................21
Testing – Using MQ Explorer and RFHTUIC .................................................................................25
Introduction

The document describes the sample COBOL MQ CICS transactions called QSUB. This transaction uses the publish/subscribe feature of WMQ V7 and requires CICS 3.2 or above.

Please note that the following PTFs need to be applied to support this transaction

CICS TS 3.2 – PK66866 (UK52671,UK52672,UK52673,UK52680) OR
CICS TS 4.1 – PK89844 (UK52619,UK52667,UK52668,UK52669)

This transaction is single purpose – all it does it subscribe to a topic and pull messages from the targeted queue. MQ messages and attributes of a process definition are used to provide the necessary information to create the subscription and to report on the status of the subscription request.

This document assumes the reader is somewhat familiar with WMQ, CICS, and COBOL. It also makes use of another ATS developed transaction, QPUB. Information about this transaction may be found at:

http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS4549

Terms

Control Message – the message used to start the QSUB transaction. It contains the basic information about the number of messages to get, the subscription queue, the MQ Topic object (optional), the MQ Topic string (optional), a correlation ID (optional) and a subscription name (optional).

Resolved topic – the resolved topic is the complete topic string used for publications. It is made up of the topic string attribute from the defined topic object, the topic string supplied in the control message when no topic object has been specified, or a concatenation of the topic string attribute of the topic object followed by a slash (‘/’) and the topic string from the control message. In the sample program, the resolved topic is limited to 400 characters.

Trigger message – this is the message passed to the processing program when the transaction is triggered. A complete description of the trigger message is documented in the WebSphere MQ InfoCenter, or for those have the older MQ manuals in the Application Programming Reference (publication number SC34-6940).
QSUB – Sample CICS transaction to subscribe to publications

Description:

The QSUB transaction executes the QSUBCBL program; it subscribes to publications based on the provided topic object, topic string or a combination of the two fields. It is started by a formatted ‘control message’ which triggers the transaction. It also uses information from the WMQ process definition.

Inputs:

The Subscription Control message.

The Subscription Control message is a free form area; commas are used to delimit the fields. The data is broken up into the following fields:

01 SUBSCRIPTION-PARMS.
  05 SUB-CONTROL PIC 9(05) VALUE 1.
  05 SUB-QUEUE PIC X(48) VALUE SPACES.
  05 TOPIC-OBJECT PIC X(48) VALUE SPACES.
  05 TOPIC-STRING PIC X(200) VALUE SPACES.
  05 MATCH-CORRELID PIC X(24) VALUE SPACES.
  05 SUB-NAME PIC X(200) VALUE SPACES.

The fields are used as follows:

- **SUB-CONTROL** is the number of messages to be retrieved from the subscription queue. This can be a range of 00001-99999.
- **SUB-QUEUE** is the name of the queue used for the publications. This sample does not allow the use of managed subscriptions, so this is a required field. The maximum length is 48 characters.
- **TOPIC-OBJECT** is the MQ defined topic object. The field is optional, if supplied, the topic string from the topic object definition is used as part (or the whole of) the resolved topic.
- **TOPIC-STRING** is an optional component of the resolved topic. If omitted and the TOPIC-OBJECT is also omitted, the SYSTEM.DEFAULT.TOPIC is used for subscription. If it is present, and the TOPIC-OBJECT is omitted, this value is used as the resolved topic for publication. If it is present and the TOPIC-OBJECT is also present, the resolved topic is created by WMQ from the topic string defined on the topic object, a concatenated slash to indicate hierarchy, and the topic string.
  - Note – if you are using a topic sting with embedded blanks, the length calculation in this sample will be incorrect.

© Copyright IBM Corporation, 2011 QSUB
• MATCH-CORRELID is an optional field, if provided:
  o It sets a fixed correlation ID on all the publications delivered as a result of
    this subscription
  o Is used as a match option for the MQGETs.
• SUB-NAME is an optional field, which may be used to name the subscription.

Sample control message:

00005,QSUB.SUB.QUEUE,QPUB.TEST, ,LYN CORRELID1,SUBNAME-LYN

The values were assigned as follows:

<table>
<thead>
<tr>
<th>Field name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUB-CONTROL</td>
<td>5</td>
</tr>
<tr>
<td>SUB-QUEUE</td>
<td>QSUB.SUB.QUEUE</td>
</tr>
<tr>
<td>TOPIC-OBJECT</td>
<td>QPUB.TEST</td>
</tr>
<tr>
<td>TOPIC-STRING</td>
<td>spaces (note a space must be left between the commas to omit a parameter)</td>
</tr>
<tr>
<td>MATCH-CORRELID</td>
<td>LYN CORRELID1</td>
</tr>
<tr>
<td>SUB-NAME</td>
<td>SUBNAME-LYN</td>
</tr>
</tbody>
</table>

The following table describes the resolved topic that will be used for message publication. It assumes that the topic object named has a topic string that contains the name, though that is not a requirement.

<table>
<thead>
<tr>
<th>TOPIC-OBJECT</th>
<th>TOPIC-STRING</th>
<th>Resolved topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blank</td>
<td>Blank</td>
<td>SYSTEM.DEFAULT.TOPIC</td>
</tr>
<tr>
<td>Fruit</td>
<td>Blank</td>
<td>Fruit</td>
</tr>
<tr>
<td>Cookware</td>
<td>Castiron/Skillet</td>
<td>Cookware/Castiron/Skillet</td>
</tr>
<tr>
<td>Blank</td>
<td>ShoeSale/Sneakers</td>
<td>ShoeSale/Sneakers</td>
</tr>
</tbody>
</table>

**The Trigger message** contains data fields that are used as follows:

• MQTM-QNAME – the name of the publication control queue, in the sample delivered it is ‘QSUB.CONTROL.QUEUE’
• MQTM-ENVDATA – this is taken from the process definition, and may be used to supply the status queue name (see Outputs). If not supplied on the process definition, this defaults to ‘QSUB.STATUS.QUEUE’.
• MQTM-USERDATA – this is taken from the process definition and may be used to supply a wait interval for the MQGET on the subscription queue.
Outputs:

The status message, which has the following layout:

```cobol
01 STATUS-MESSAGE.
   05 FILLER PIC X(20)
      VALUE 'SUBSCRIPTION MSGS= '.
   05 SM-NUMBER PIC 9(5) VALUE ZEROS.
   05 FILLER PIC X(20)
      VALUE ' FOR TOPIC = '.
   05 SM-TOPIC PIC X(48) VALUE SPACES.
   05 FILLER PIC X(20)
      VALUE ' FOR RESOLVED = '.
   05 SM-RESOLVED PIC X(400) VALUE SPACES.
```

The fields are used as follows:

SM-NUMBER is the total number of messages retrieved during the subscription period.
SM-TOPIC – the topic object, if provided, from the subscription control message.
SM-RESOLVED – this is the resolved topic string used for the subscription.
**QSUBCBL Program Flow:**

1) The QSUB transaction is triggered.

2) The control queue is opened.

3) Subscription control message is read.

4) Control message is parsed into the controlling fields.

5) The subscription request is built and submitted.

6) The subscription destination queue is opened.

7) Messages are read from the queue in a loop, until the control count has been reached or the wait time.

8) The status message is built.

9) The status queue opened and the status message is put.

10) All queues are closed.

11) Control returns to CICS.

Important note: The subscription is not closed until the transaction ends, during testing you may see published messages on the subscription destination queue that have arrived after the queue was closed but before the transaction ended. This is because of the structure of the program and the nature of pub/sub. Using message expiration and clean up either within application or with another program or tool should be considered when creating pub/sub applications.

If testing with a resolved topic string of longer than 400 characters is required, the size can be adjusted in the program QSUBCBL and the program compiled and linked. The following fields may need to be altered to support a larger area:

- **TOPIC-STR** – 200 byte working storage field that has a copy of the input topic string from the subscription control message.
- **RESOLVED-TOPIC-STR** – 400 byte working storage field that is used to pass to WMQ to construct the resolved topic string from the topic object and topic string from the subscription control message.
- **TOPIC-STRING** – 200 byte working storage field that is part of the subscription control message.
- **SM-RESOLVED** - 400 byte working storage field that is part of the status message, it gives the resolved topic string returned from WMQ.
CICS Definitions:

There are two CICS definitions required, one for the QSUB transaction and one for the QSUBCBL COBOL program. The source for the COBOL program QSUBCBL and the CICS definitions are included. If the names need to be altered to fit corporate standards, the MQ process definition will also need to be changed to point to the correct transaction ID.

QSUB definition:

Using CEDA the QSUB definition was created using the following steps:

1) Enter the CEDA defining the transaction and the associated program to a specified group; in the example the group QML0GRP is used.

```
ceda def trans(qsub) group(qml0grp)
```
2) The definition success message should appear as shown. Note that the TASKDATALoc can be ‘Any’.

```cobol
DEF TRANS(QSUB) GROUP(QML0GRP)
OVERTYPE TO MODIFY CICS RELEASE = 0660
CEDA DEFINE TRANSACTION( QSUB )
TRANSaction ==> QSUB
Group ==> QML0GRP
DESCRIPTION ==>
PROgram ==> QSUBCBL
TWASIZE ==> 00000 0-32767
PROFILE ==> DFHICIST
PARTITIONSET ==>
STATUS ==> Enabled Enabled | Disabled
PRIMESIZE : 00000 0-65520
TASKDATALOC ==> Any Below | Any
TASKDATAKEY ==> User User | Cics
STORAGECLEAR ==> No No | Yes
RUNAWAY ==> System System | 0 | 500-2709000
SHUTDOWN ==> Disabled Disabled | Enabled
ISOLATE ==> Yes Yes | No
BREXIT ==>
```

* REMOTE ATTRIBUTES
**QSUBCBL Definition:**

1) Enter the define program command, supplying the group name.

```
DEF PROG(QSUBCBL) GROUP(QML0GRP)
```

2) Set the language to COBOL, the data location to any and hit the enter key. The ‘Define Successful’ message should be displayed as follows.

```
DEF PROG(QSUBCBL) GROUP(QML0GRP)
OVERTYPE TO MODIFY CICS RELEASE = 0660
CEDA DEFINE PROGRAM(QSUBCBL)
PROGRAM : QSUBCBL
Group : QML0GRP
DESCRIPTION => Sample MQ Subscription program
Language => COBOL | Assembler | Le370 | C | Pl1
REload => No | Yes
Resident => No | Yes
Usage => Normal | Transient
USElpack => No | Yes
Status => Enabled | Disabled
RSl => 00 | 0-24 | Public
CEDF => Yes | No
DATATLOCATION => Any | Below | Any
EXECKEY => User | Cics
CONCURRENCY => Quasirent | Threaddse
API => Cicsapi | Openapi
REMOTE ATTRIBUTES
* DYNAmic => No | Yes
```

**QML0GRP Installation**

The new resources must be installed into the region.

1) Enter the CEDA install command as shown:

```
CEDA IN G(QML0GRP)
```
2) The install successful messages should be displayed as shown:

```
IN G(QMLGRP)
OVERTYPE TO MODIFY
CEDA  Install
   ALL
     ATomservice   ==>
     Bundle       ==>
     CONNection   ==>
     CORbaserver  ==>
     DB2Conn      ==>
     DB2Entry     ==>
     DB2Tran      ==>
     DJar         ==>
     DOctemplate  ==>
     Enqmodel     ==>
     File         ==>
     Ipconn       ==>
     J0urnalmodel ==>
     JVMserver    ==>
     LIBrary      ==>
   +  LSrpool     ==>
```

INSTALLED SUCCESSFUL

SYSID=TOR1 APPLID=CTSTOR01
TIME: 12.00.21 DATE: 01/28/11
WMQ Definitions:

Process Definition, QSUB.PROCESS

To use triggering to initiate the QSUB transaction, a process definition is needed. This definition includes the CICS transaction name, if QSUB is not used as the transaction name this needs to be altered to reflect that. The definition also includes User and Environment data fields that QSUBCBL uses to supply information about the get wait interval (10 seconds in the example shown) and the status queue. If the supplied values are not correct for the environment being used, please alter them as necessary.

This process definition sample is included with this TechDoc as QSUB_PROCESS_DEF.txt. From MQ Explorer the process definition is shown below.

<table>
<thead>
<tr>
<th>Field</th>
<th>Meaning</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application ID</td>
<td>CICS TX ID</td>
<td>QSUB</td>
</tr>
<tr>
<td>Environment data</td>
<td>Output Status queue</td>
<td>QSUB.STATUS.QUEUE</td>
</tr>
<tr>
<td>User data</td>
<td>Get Wait Milliseconds</td>
<td>10000</td>
</tr>
</tbody>
</table>
Queue Definitions

Three queues are needed for testing the QSUB transaction. The samples are defined as follows:

- QSUB.CONTROL.QUEUE – the publication control queue. This queue is triggered and initiates the publication process. If changes have been made to the CICS transaction name or the CICS initiation queue for the region where the QSUB transaction will run, those values need to be changed before defining this queue.
- QSUB.STATUS.QUEUE – this publication status queue. This queue holds status messages, generated after the subscription process has ended.
- QSUB.SUB.QUEUE – the subscription queue. This queue is the subscription destination queue, used for testing that the subscription process is working.

Sample queue definitions are included with this TechDoc as QSUB_QUEUE_DEF.txt. A display of the MQ Explorer definitions of each queue and the necessary attributes for this sample are shown on the following pages. Note that the required fields are listed in a table below the picture, the other fields have default values.
2) QSUB.CONTROL.QUEUE

The basic definition:

Required Fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queue Name</td>
<td>QSUB.CONTROL.QUEUE</td>
</tr>
</tbody>
</table>
The triggering information for QSUB.CONTROL.QUEUE:

![Properties window for QSUB.CONTROL.QUEUE](image)

Required Fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigger control</td>
<td>On</td>
</tr>
<tr>
<td>Trigger type</td>
<td>First</td>
</tr>
<tr>
<td>Initiation queue</td>
<td>CICS.CTSTOR02.INITQ</td>
</tr>
<tr>
<td>Process name</td>
<td>QSUB.PROCESS</td>
</tr>
</tbody>
</table>
3) QSUB.STATUS.QUEUE

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queue Name</td>
<td>QSUB.STATUS.QUEUE</td>
</tr>
</tbody>
</table>
4) QSUB.SUB.QUEUE

5) Required Fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queue Name</td>
<td>QSUB.SUB.QUEUE</td>
</tr>
</tbody>
</table>
### Topic Object Definition, QPUB.TEST

A sample topic definition QPUB.TEST is delivered. The topic string associated with this object is ‘QPUB’.

This sample is included with the TechDoc as QSUB_TOPIC_DEF.txt.

From the MQ Explorer the topic object looks as shown below.

![MQ Explorer screenshot](image)

6) **Required Fields:**

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topic Name</td>
<td>QSUB.TEST</td>
</tr>
<tr>
<td>Topic string</td>
<td>QSUB</td>
</tr>
<tr>
<td>Description (optional)</td>
<td>QSUB TRANSACTION SAMPLE TOPIC</td>
</tr>
</tbody>
</table>
Testing Samples

Four test subscription and two test publication sample messages are delivered with this TechDoc. Each was created to test a specific subscription type. They can be used with any tool that can read a file and put the message to a queue. Internally we used a combination of OEMPUTX (supplied in SupportPac IP13), the MQ Explorer, and RFHUTIL (supplied in SupportPac IH03). The QPUB transaction, delivered in TechDoc PRS4549 was used as the publication engine.

Please note that when testing the get wait interval, supplied on the process definition or in the QSUBCBBL program, may have to be lengthened to get results. In the testing done as part of this program development, CEDX was used to stop the subscription program following the successful subscription and the publication transaction was initiated at that point. This was done for timing reasons; when working with the panels often the subscription would end before the publication could get started and vice versa.

Also, the number of publications on the publication control messages is typically more than the number on subscription control. That was done as a convenience, as during testing it was often beneficial to run the subscription test a number of times.

Subscription sample messages and test information

<table>
<thead>
<tr>
<th>File Name</th>
<th>Test Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>QSUB_TOPICOBJECT.txt</td>
<td>Specifies the topic object and a subscription name.</td>
</tr>
<tr>
<td>QSUB_TOPICSTRING.txt</td>
<td>Specifies the topic object and a topic string, along with a subscription name.</td>
</tr>
<tr>
<td>QSUB_CORRELID_TEST1A.txt</td>
<td>Specifies the topic object, a correlation ID, and a subscription name.</td>
</tr>
<tr>
<td>QSUB_CORRELID_TEST1B.txt</td>
<td>Specifies the topic object and topic string, a correlation ID and a subscription name.</td>
</tr>
</tbody>
</table>
Publication sample messages and test information

<table>
<thead>
<tr>
<th>File Name</th>
<th>Test information</th>
</tr>
</thead>
<tbody>
<tr>
<td>QPUB_TOPICOBJECT.txt</td>
<td>Publishes 2005 messages to the defined topic object. Note this file is fixed length and has a great deal of white space with must be included in the publication control message. This can be used with the QSUB_TOPICOBJECT and QSUB_CORRELID_TEST1A subscription tests.</td>
</tr>
<tr>
<td>QPUB_TOPICSTRING.txt</td>
<td>Publishes 2005 messages to the defined topic object and topic string combination. Note this file is fixed length and has a great deal of white space with must be included in the publication control message. This can be used with the QSUB_TOPICSTRING and QSUB_CORRELID_TEST1B subscription tests.</td>
</tr>
</tbody>
</table>

Note: The test files described above should be uploaded to z/OS. The subscription test messages have a fixed length of 70 bytes; the publication test messages are also fixed length at 301 bytes.
Testing – Batch Test Submission

1) Testing using OEMPUTX (from SupportPac IP13) is the simplest method on z/OS. Sample JCL is delivered with this document to execute two instances of the test program. The first instance submits the subscription control message and awaits the reply on the subscription status queue, the second submits the publication control message and awaits the status message on the appropriate queue.

2) Sample JCL for the batch test is included in the OEMPUTX_JCL.txt file that is included with this TechNote.

3) The status message from the QSUB step will look something like what is shown:

```
00205,QSUB.SUB.QUEUE,QPUB.TEST,PEACHES,QSUB CORRELID1B,SUBNAME-QSUB1B
```

4) Output from the QSUB step should look something like what is shown below, note this includes both the input and status messages:

```
buffer:-gQSUB.CONTROL.QUEUE * request queue
buffer:-rQSUB.STATUS.QUEUE * reply queue
buffer:-fileDD:MSGIN
buffer:-dJTEST4
parm: -mQML0 -n1 -x -pm
Message file: DD:MSGIN
OEMPUTX about to MQCONN to QMgr QML0.
OEMPUTX about to MQOPEN request queue: QSUB.CONTROL.QUEUE
OEMPUTX about to MQOPEN reply queue: QSUB.STATUS.QUEUE
CPU type 0000012094
Date Time 2011/07/26 03:24:06.
Description JTEST4.
Entering PUT/GET loops (MQGET_Wait=60 seconds)...
Preload the queue with 0 messages...
Message size : <Determined by Message File data>
Message persistence : NON-PERSISTENT
Messages per loop : 1
Total messages : 1
Syncpoints : NO-SYNCPPOINT
MQGET replies by : Any message
Starting loop at 2011-07-26 03:24:06.647410
```
<<< MESSAGE PUT TO REQUEST QUEUE ** 1 ** >>>>
MQMD header size is 324 bytes
StrucId :MD   Version :1
Report :00000000   MsgType :00000008
Expiry :FFFFFFFF   Feedback :00000000
Encoding :00000311   CodedCharSetId :00000000 (0)
Format :MQSTR
Priority :FFFFFFFF   Persistence :00000000
MsgId :CSQ QML0 H..9q...
<CED4DDDF44444444C20F9235>
<328084300000000080B98DC4>
CorrelId :........................
<000000000000000000000000>
<000000000000000000000000>
BackoutCount :00000000
ReplyToQ :QSUB.STATUS.QUEUE
ReplyToQMgr :
UserIdentifier :ELKINSC
AccountingToken
<08D5D6C9C5C6E4D1C90000000000000000000000000000000000000000000000>
ApplIdentityData:
PutApplType :00000002
PutApplName :ELKINSC1
PutDate :20110726   PutTime :03240664
ApplOriginData :
** MESSAGE DATA (70 of 70 bytes printed) **
00000000 : F0F0F2F0 F56BD8E2 E4C24BE2 E4C24BD8    00205,QSUB.SUB.Q
00000010 : E4C5E4C5 6BD8D7E4 C24BE3C5 E2866BD7    UEUE,QPUB.TEST,P
00000020 : C5C1C3C8 C5E26BD8 E2E4C240 C3D6D9D9    EACHES,QSUB CORR
00000030 : C5D3C9C4 F1C26BE2 E4C2D5C1 D4C560D8    ELID1B,SUBNAME-Q
00000040 : E2E4C2F1 C240
SUB1B
<<< END OF MESSAGE PUT TO REQUEST QUEUE ** 1 ** >>>>
<<< MESSAGE GOT FROM REPLY QUEUE ** 1 ** >>>>
MQMD header size is 324 bytes

MQMD header size is 324 bytes

StrucId :MD Version :1
Report  :00000000 MsgType :00000008
Expiry  :FFFFFFFF Feedback :00000000
Encoding :0000311 CodedCharSetId :000001F4 (500)
Format  :MQSTR
Priority :00000000 Persistence :00000000
MsgId   :CSQ QML0 H.....%.
CorrelId :CSQ QML0 H.......  
BackoutCount :00000000
ReplyToQ :QPUB.PUB.STATUS
ReplyToQMgr :QML0
UserIdentifer :CICSUSER
AccountingToken

** MESSAGE DATA (513 of 513 bytes printed) **

00000000 : E2E4C2E2 C3D9C9D7 E3C9D6D5 40D4E2C7 SUBSCRIPTION MSG
00000010 : E27E4040 F00F2F0 F540C6D6 D940E3D6 S= 00205 FOR TO
00000020 : D7C9C340 7E404040 40404040 40D8D7E4 PIC = QPU
00000030 : C24BE3C5 E2E34040 40404040 40404040 B.TEST
00000040 : 40404040 40404040 40404040 40404040
00000050 : 40404040 40404040 40404040 4040C6D6 QPU
00000060 : D940D9C5 E2D6D3E5 C5C4407E 40404040 R RESOLVED =
00000070 : 40D8D7E4 C261D7C5 C1C3C8C5 E2404040 PUB/PEACHES
00000080 : 40404040 40404040 40404040 40404040
00000090 : 40404040 40404040 40404040 40404040
000000A0 : 40404040 40404040 40404040

----- All Blanks -----

00000100 : 40404040 40404040 40404040 40404040
00000110 : 40404040 40404040 40404040 40404040
00000120 : 40404040 40404040 40404040 40404040
00000130 : 40404040 40404040 40404040 40404040
00000140 : 40404040 40404040 40404040 40404040
00000150 : 40404040 40404040 40404040 40404040
00000160 : 40404040 40404040 40404040 40404040
00000170 : 40

<<<< END OF MESSAGE GOT FROM REPLY QUEUE ** 1 ** >>>>>
Message Data (ASCII->EBCDIC converted) from 500 to 0

00000000 : EBCF29EB 06233576 ED35750A 7C74EB66 ............@...
00000010 : EBA17C7C CACA1BCA B57C4675 237CED75 ~@@.....@...@
00000020 : 7635067C A17C7C7C 7C7C7C7C 7C7776CF ...~@@@@@@@@...
00000030 : 29D2ED2A EBE07C7C 7C7C7C7C 7C7C7C7C .K....@@@@@@@@
00000040 : 7C7C7C7C 7C7C7C7C 7C7C7C7C 7C7C7C7C @@@@@@@@@@@@@@@
00000050 : 7C7C7C7C 7C7C7C7C 7C7C7C7C 7C7C7C7C @@@@@@@@@@@@@@@
00000060 : 7C7C7C7C 7C7C7C7C 7C7C7C7C 7C7C7C7C @@@@@@@@@@@@@@@
00000070 : 7C7C7C7C 7C7C7C7C 7C7C7C7C 7C7C7C7C @@@@@@@@@@@@@@@
00000080 : 7C7C7C7C 7C7C7C7C 7C7C7C7C 7C7C7C7C @@@@@@@@@@@@@@@
00000090 : 7C7C7C7C 7C7C7C7C 7C7C7C7C 7C7C7C7C @@@@@@@@@@@@@@@
000000A0 : 7C7C7C7C 7C7C7C7C 7C7C7C7C 7C7C7C7C @@@@@@@@@@@@@@@
000000B0 : 7C7C7C7C 7C7C7C7C 7C7C7C7C 7C7C7C7C @@@@@@@@@@@@@@@
000000C0 : 7C7C7C7C 7C7C7C7C 7C7C7C7C 7C7C7C7C @@@@@@@@@@@@@@@
000000D0 : 7C7C7C7C 7C7C7C7C 7C7C7C7C 7C7C7C7C @@@@@@@@@@@@@@@
000000E0 : 7C7C7C7C 7C7C7C7C 7C7C7C7C 7C7C7C7C @@@@@@@@@@@@@@@
000000F0 : 7C7C7C7C 7C7C7C7C 7C7C7C7C 7C7C7C7C @@@@@@@@@@@@@@@
00000100 : 7C7C7C7C 7C7C7C7C 7C7C7C7C 7C7C7C7C @@@@@@@@@@@@@@@
00000110 : 7C7C7C7C 7C7C7C7C 7C7C7C7C 7C7C7C7C @@@@@@@@@@@@@@@
00000120 : 7C7C7C7C 7C7C7C7C 7C7C7C7C 7C7C7C7C @@@@@@@@@@@@@@@
00000130 : 7C7C7C7C 7C7C7C7C 7C7C7C7C 7C7C7C7C @@@@@@@@@@@@@@@
00000140 : 7C7C7C7C 7C7C7C7C 7C7C7C7C 7C7C7C7C @@@@@@@@@@@@@@@
00000150 : 7C7C7C7C 7C7C7C7C 7C7C7C7C 7C7C7C7C @@@@@@@@@@@@@@@
00000160 : 7C7C7C7C 7C7C7C7C 7C7C7C7C 7C7C7C7C @@@@@@@@@@@@@@@
00000170 : 7C7C7C7C 7C7C7C7C 7C7C7C7C 7C7C7C7C @@@@@@@@@@@@@@@
00000180 : 7C7C7C7C 7C7C7C7C 7C7C7C7C 7C7C7C7C @@@@@@@@@@@@@@@
00000190 : 7C7C7C7C 7C7C7C7C 7C7C7C7C 7C7C7C7C @@@@@@@@@@@@@@@
000001A0 : 7C7C7C7C 7C7C7C7C 7C7C7C7C 7C7C7C7C @@@@@@@@@@@@@@@
000001B0 : 7C7C7C7C 7C7C7C7C 7C7C7C7C 7C7C7C7C @@@@@@@@@@@@@@@
000001C0 : 7C7C7C7C 7C7C7C7C 7C7C7C7C 7C7C7C7C @@@@@@@@@@@@@@@
000001D0 : 7C7C7C7C 7C7C7C7C 7C7C7C7C 7C7C7C7C @@@@@@@@@@@@@@@
000001E0 : 7C7C7C7C 7C7C7C7C 7C7C7C7C 7C7C7C7C @@@@@@@@@@@@@@@
000001F0 : 7C7C7C7C 7C7C7C7C 7C7C7C7C 7C7C7C7C @@@@@@@@@@@@@@@
00000200 : 7C7C7C7C 7C7C7C7C 7C7C7C7C 7C7C7C7C @@@@@@@@@@@@@@@

---- All @ signs ---

Workload manager data

Samples %idle %unknown (MQ?) %using CPU %doing I/O

%Wait for CPU

<table>
<thead>
<tr>
<th>Workload Manager</th>
<th>%idle</th>
<th>%unknown</th>
<th>%using CPU</th>
<th>%doing I/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>QML0CHIN.0072</td>
<td>34</td>
<td>100</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>QML0MSTR.0070</td>
<td>34</td>
<td>100</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Total Transactions : 1
Elapsed Time : 0.621 seconds
Application CPU Time: 0.003 seconds (0.4%)
Transaction Rate : 1.610 trans/sec

Round trip per msg : 621226 microseconds
Avg App CPU per msg : 2612 microseconds

Jobname.ASID TCB(uS) SRB(uS) Tot(uS) (%)
/time    /time    /time
QML0MSTR.0070  00000271  00001100  00001371  0.2
QML0CHIN.0072  00000040  00000034  00000074  0.0
QML0BRK*      00000000  00000000  00000000  0.0
Total CPUmicrosecs/tran  1445

Ending loop at 2011-07-26 03:24:07.270440
OEMPUTX Normal Exit: End of program
Exiting at 2011-07-26 03:24:07.275922

© Copyright IBM Corporation, 2011 QSUB
Testing – Using MQ Explorer and RFHTUICL

1. A combination of RFHUTICL and MQ Explorer was used for this test. The subscription control message is going to be input via RFHUTICL. The initial panel will look something like what is shown:

5) Click on ‘Open file’ and browse to file the sample messages. Select the subscription test file, like what is shown.
6) Click on the data tab to display the contents as shown.
7) Select the ‘MQMD’ tab, and set the message format to MQSTR. If you do not set the message format the subscription will result in a 2110 return code (format error).
8) Returning to ‘Main’ tab, and click on ‘Write queue’ to put the message.

9) To use the sample publish tests, it’s a good idea to have that put ‘primed’. For our tests we generally used the MQ Explorer for the publish control message input. To do this, right click on the queue name in the list and select ‘Put Test Message’ (as shown).

10) Copy the contents of the appropriate test file, in this case QPUB_TOPICSTRING.txt into the put message buffer. Please make sure the entire message (including all trailing blanks) are included, the QPUB transaction is expect a fixed length message. When the message is pasted into the buffer, it may look empty.

11) Check the status queues to make certain that both the publication and subscription worked, there should be a message on both the publication and subscription status queues. Depending on the speed of the system, the subscription count may not be
what you expect. There may also be ‘extra’ publications on the subscription destination queue. Those messages were published after the subscription queue was closed, but prior to the subscription being ended.

The message on the publication status queue should look something like this (RFHUTILC used to display).
12) The message on the subscription status queue should look something like is shown:

![Message on subscription status queue]

13) The ‘extra publications’ can also be viewed.
Acknowledgements:

The authors would like to thank the following people for their assistance:
Mark Taylor
Shalawn King
Jenifer Foley
Chris Griego
Ashley Curry