MQ Pub/Sub: C API and traces

http://www-01.ibm.com/support/docview.wss?uid=swg27050203

Angel Rivera (rivera@us.ibm.com)
IBM MQ Distributed Level 2 Support
Date last updated: 20-Sep-2017

Link to index: https://developer.ibm.com/answers/questions/402074/mq-pubsub-training-presentations.html

Related presentations

This presentation is a continuation of:

http://www.ibm.com/support/docview.wss?uid=swg27050138
MQ Pub/Sub: non-durable topics and subscribers

http://www.ibm.com/support/docview.wss?uid=swg27050162
MQ Pub/Sub: topic tree, security

http://www.ibm.com/support/docview.wss?uid=swg27050181
MQ Pub/Sub: durable subscribers

This presentation is one of a series. For the complete list, please see:

https://developer.ibm.com/answers/questions/402074/mq-pubsub-training-presentations.html
MQ Pub/Sub: training presentations

Related zip files

This techdoc has 2 zip files with files that are discussed in this presentation:

- QMPS-pub-sub-C-API-original-trace.zip
- QMPS-pub-sub-C-API-Application-Activity-Trace.zip

Agenda

- This presentation examines the C API from the following MQ samples (local bindings):
- amqspub => publish a message
- amqssub => non-durable subscriber
- amqssbx => durable subscriber

- “Normal” traces were obtained from running the samples and are explained.
- The Application Activity Trace was obtained too.

Scenario

- Scenario consists of:
- 1 publisher publishing 1 message to a topic
- While 2 subscribers receive the message.
- one subscriber is durable
- another subscriber is non-durable

- 4 command prompt windows were used:

Window1 (admin)
Window2 (durable sub)
Window3 (non durable sub)
Window4 (pub)

Scenario

- Window1 (admin): enable activity trace;
  normal trace: **strmqtrc -m QMPS -t all -t detail**

- Window2 (durable sub):
  amqssbx -m QMPS -d SUB20 -q Q3 -t sales -k

- Window3 (non durable sub):
  amqssub sales QMPS

- Window4 (pub): **amqspub sales QMPS**

- Window1 (admin): disable activity trace
  end normal trace: **endmqtrc -a**

Application Activity Trace

I am just taking advantage that the MQ traces are going to be discussed in this presentation, and I wanted to talk briefly about a related feature:

Application Activity Trace

Even though it does not provide any specific value added exclusively for Pub/Sub, it is still useful to see the activity of the MQ samples.

Application Activity Trace

1) The following has an excellent practical introduction on this feature:
http://www.ibm.com/developerworks/websphere/library/techarticles/1306_bushby/1306_bushby.html
Increasing the visibility of messages using WebSphere MQ Application Activity Trace

2) This technote complements the above article with very practical information:
http://www.ibm.com/support/docview.wss?uid=swg21669530
How to get activity trace only for selected applications

Notes: Application Activity trace

notes

- Step 1) Configure the mqat.ini file.
- The configuration file **mqat.ini** enables you to control the frequency and level of detail in Application Activity Trace.
- The mqat.ini file is located in the queue manager data directory:
- On Linux® and UNIX®: /var/mqm/qmgrs/<qm_name>
- On Microsoft® Windows®: C:\Program Files\IBM\WebSphere MQ\qmgrs\<qm_name>

- Step 2) You will need to recycle the queue manager or

- You can use runmqsc to disable and then to enable the activity trace:
- To disable the activity trace: ALTER QMGR ACTVTRC(OFF)
- To Enable the activity trace: ALTER QMGR ACTVTRC(ON)

- From the Explorer:
- Queue Manager > Properties > Online monitoring > Activity trace

Notes: Application Activity trace

notes

- Step 3) Perform the scenario (aka "activity").

- Step 4) Use amqsact to see the activity trace
  amqsact -m QMPS -b

- Notes:
- The **flag -b is for browse** and does NOT destroy the activity records. This is OK when you are experimenting with amqsact.
- However, once you get familiar with amqsact it is recommended that you do not use the -b flag in order to consume (destructive get) the corresponding records.

Location of Samples

Windows:

Source: %MQ_INSTALLATION_PATH%\Tools\c\Samples

Exec: %MQ_INSTALLATION_PATH%\Tools\c\Samples\bin64

Unix: Needs fileset for the MQ Samples.

Source: $MQ_INSTALLATION_PATH/samp

Executable: $MQ_INSTALLATION_PATH/samp/bin

There are 2 variations on the executables:

- local bindings: **amqspub**
- client mode (filename ends with ‘c’): **amqspubc**

Notes: Source amqspuba.c

notes

- Comments from the source code that are relevant in this presentation.

```
/*   AMQSPUBA is a sample C program to publish messages on a topic  */
/*   and is an example of the use of MQPUT.                         */

/*   AMQSPUBA has the following parameters                          */
/*       required:                                                  */
/*                 (1) The name of the target topic                 */
/*       optional:                                                  */
/*                 (2) Queue manager name                           */
/*                 (3) The publish options
```

```
/*********************************************************************/
/* Extract arguments                                                 */
/*  argv[1] - Topic String                                           */
/*  argv[2] - Queue Manager name (optional)                          */
/*  argv[3] - Publish options (optional)                             */
/*            e.g. 2097152 = MQPMO_RETAIN - Retain publication       */
/*********************************************************************/
```

Notes: Specifying publish options (part 1)

notes

- Question:
- How to specify the publish options to amqspub?

- Answer:
- As a decimal integer.
- The source code for amqspub has an example:
- /* e.g. 2097152 = MQPMO_RETAIN - Retain publication */

- You cannot specify in the command line the name of the corresponding constant: MQPMO_RETAIN
- .
- Then, how do you find out which is the number associated with this constant?
- Look at the **header include file cmqc.h**
- It shows the Hexadecimal value
- #define MQPMO_RETAIN 0x00200000

Notes: Specifying publish options (part 2)

notes

- You cannot use this hexadecimal value in the invocation of amqspub (it will not be interpreted properly):
- C:> amqspub sports QMPS 0x00200000
- Sample AMQSPUBA start
- publish options are **0**
- target topic is sports

- You need to convert the hexadecimal to decimal
- You could use an online converter, such as:
- http://www.binaryhexconverter.com/decimal-to-hex-converter
- The corresponding decimal value is:
- 2097152
- .
- C:\>amqspub sports QMPS 2097152
- Sample AMQSPUBA start
- publish options are 2097152
- target topic is sports

Notes: Source amqspuba.c

notes

+ The code uses the traditional verbs and sequence:

```
MQCONN to queue manager
     MQOPEN topic
          while { handle messages
               MQPUT => publish message
          }
     MQCLOSE topic
MQDISC from queue manager
```

+ Highlights:

MQOPEN is used to “open” a Topic.
MQPUT is used to Publish a message
Note: there is no MQPUB verb!!

Notes: Source amqspuba.c

notes

+ The following code is used to handle topic strings.
The specification of the topic string uses a variable-length string (see next slide for more details).

```
/* Use parameter as the name of the target topic */
   od.ObjectString.VSPtr=argv[1];
   od.ObjectString.VSLength=(MQLONG)strlen(argv[1]);

   printf("target topic is %s\n", (char*)od.ObjectString.VSPtr);

/*   Open the target topic for output  */
   od.ObjectType = MQOT_TOPIC;
   od.Version = MQOD_VERSION_4;
   MQOPEN(Hcon, …
```

Notes: Source amqspuba.c

notes

Variable-length strings are used for topic strings.
These are represented as the data type called MQCHARV.

For a good explanation on MQCHARV see the free redbook:
http://www.redbooks.ibm.com/abstracts/SG247583.html?Open
WebSphere MQ V7.0 Features and Enhancements (SG24-7583)

Chapter 6: Message Queue Interface extensions
Section 6.1: Variable-length strings
Pages 86-88

The maximum size of a topic string is 10,240 bytes (quite large!)
Note: From the header include file “cmqc.h”

```
#define MQ_TOPIC_NAME_LENGTH        48    <<= Topic Object name
#define MQ_TOPIC_STR_LENGTH      10240    <<= Topic string limit
```

Notes: Normal trace for amqspub

notes

Original file name: AMQ31677.0.FMT
Renamed file name: AMQ31677.amqspub.FMT
Command: amqspub sales QMPS

```
.
 08:01:44.488778    31677.1      CONN:5400006 {  MQOPEN
...
# The topic string is: sales
08:01:44.488842    31677.1      CONN:5400006     ObjectString:
 08:01:44.488843    31677.1      CONN:5400006       0x0000:  73616c65 73                |sales       |
...
 08:01:44.489892    31677.1      CONN:5400006 }  MQOPEN rc=OK FunctionTime=1114


*08:01:48.913347    31677.1      CONN:5400006 {  MQPUT
 08:01:48.913403    31677.1      CONN:5400006      No RFH2 format properties in message
 08:01:48.913407    31677.1      CONN:5400006      MQPUT to hObj:2 ObjectType:8 ObjectName:: ResObjName::
 08:01:48.913410    31677.1      CONN:5400006      __________
 08:01:48.913412    31677.1      CONN:5400006      MQPUT >>
...
# The published message text is: TEST-PUB
 08:01:48.913440    31677.1      CONN:5400006     Buffer:
 08:01:48.913442    31677.1      CONN:5400006       0x0000:  54455354 2d505542          |TEST-PUB    |
...
 08:01:48.938651    31677.1      CONN:5400006 }  MQPUT rc=OK FunctionTime=25304
.
```

Notes: Activity Trace for amqspub (part 1)

notes

Original file name: QMPS-activity-trace-1.txt

Notes:
There is no information on the topic string, nor on the payload.
Time granularity is seconds (not milliseconds)

```
MonitoringType: MQI Activity Trace
Correl_id:
00000000:  414D 5143 514D 5053 2020 2020 2020 2020  'AMQCQMPS        '
00000010:  8931 9C59 7579 0420                      '.1.Yuy.         '
QueueManager: 'QMPS'
Host Name: 'mosquito'
IntervalStartDate: '2017-08-23'
IntervalStartTime: '08:01:44'
IntervalEndDate: '2017-08-23'
IntervalEndTime: '08:03:03'
CommandLevel: 903
SeqNumber: 0
ApplicationName: 'amqspub'
Application Type: MQAT_UNIX
ApplicationPid: 31677
UserId: 'mqm'
```

Notes: Activity Trace for amqspub (part 2)

notes

```
API Caller Type: MQXACT_EXTERNAL
API Environment: MQXE_OTHER
Application Function: ''
Appl Function Type: MQFUN_TYPE_UNKNOWN
Trace Detail Level: 3
Trace Data Length: 300
Pointer size: 8
Platform: MQPL_UNIX
```

Tid	Date	Time	Operation	CompCode	MQRC	HObj
001	2017-08-23	08:01:44	MQXF_CONN	MQCC_OK	0000	-
001	2017-08-23	08:01:44	MQXF_OPEN	MQCC_OK	0000	2
001	**2017-08-23**	**08:01:48**	**MQXF_PUT**	**MQCC_OK**	**0000**	**2**
001	2017-08-23	08:03:03	MQXF_CLOSE	MQCC_OK	0000	2
001	2017-08-23	08:03:03	MQXF_DISC	MQCC_OK	0000	-

Notes: Source amqssuba.c

notes

- Comments from the source code that are relevant in this presentation.

```
/* Description: Sample C program that subscribes and gets messages  */
/*              from a topic (example using MQSUB). A managed       */
/*              destination queue is used.                          */

/*   AMQSSUBA has the following parameters                          */
/*       required:                                                  */
/*                  (1) The name of the topic                       */
/*       optional:                                                  */
/*                  (2) Queue manager name                          */
/*                  (3) The MQSD.Options to pass into MQSUB         */
```

Notes: Source amqssuba.c

notes

+ The code uses the MQSUB verb instead of MQOPEN:

```
MQCONN to queue manager
     MQSUB topic
          while { handle messages
               MQGET => get message
          }
     MQCLOSE topic
MQDISC from queue manager
```

+ Highlights:

MQSUB is used to “open” a subscription! (and not MQOPEN)
MQGET is used to get the published messages

Notes: Source amqssuba.c

notes

+ The following code is used for MQSUB (the topic string uses a variable-length string, as explained with amqspuba.c)

```
/*   Subscribe using a managed destination queue     */
sd.Options =   MQSO_CREATE
             | MQSO_NON_DURABLE
             | MQSO_FAIL_IF_QUIESCING
             | MQSO_MANAGED;

if (argc > 3)
{
  sd.Options = atoi( argv[3] );
  printf("MQSUB SD.Options are %d\n", sd.Options);
}

sd.ObjectString.VSPtr = argv[1];
sd.ObjectString.VSLength = (MQLONG)strlen(argv[1]);
MQSUB(Hcon, …
```

Notes: Normal trace for amqssub

notes

Original file name: AMQ31675.0.FMT
Rename file name: AMQ31675.amqssub.FMT
Command: amqssub sales QMPS

.

```
08:01:42.960534    31675.1      CONN:5400006 {  MQSUB

# The topic string is: sales
08:01:42.960579    31675.1      CONN:5400006      ObjectString:
08:01:42.960580    31675.1      CONN:5400006        0x0000:  73616c65 73                |sales          |
.
# Need to know the “managed” queue (Subscription-Queue)
# It is: 'SYSTEM.MANAGED.NDURABLE.599C318920047671'
08:01:42.963825    31675.1      CONN:5400006      MQINQ <<
…
08:01:42.963831    31675.1      CONN:5400006      Hobj:
08:01:42.963832    31675.1      CONN:5400006        0x0000:  04000000                  |....           |
08:01:42.963833    31675.1      CONN:5400006        ObjHdl:4 ObjType:SUBSCRIPTION-QUEUE
…
08:01:42.963840    31675.1      CONN:5400006      Charattrs:
 08:01:42.963841    31675.1      CONN:5400006        0x0000:  53595354 454d2e4d 414e4147 45442e4e  |SYSTEM.MANAGED.N|
 08:01:42.963841    31675.1      CONN:5400006        0x0010:  44555241 424c452e 35393943 33313839  |DURABLE.599C3189|
 08:01:42.963841    31675.1      CONN:5400006        0x0020:  32303034 37363731 20202020 20202020  |20047671        |
08:01:42.963849    31675.1      CONN:5400006 ----}  MQINQ rc=OK FunctionTime=673
 08:01:42.963852    31675.1      CONN:5400006 ----{  smqopGetPolicy
 08:01:42.963854    31675.1      CONN:5400006      /build/slot1/p900_P/src/lib/ams/smqoplca.c : 164  parameters
'QMPS' 'SYSTEM.MANAGED.NDURABLE.599C318920047671      '
```

Notes: Normal trace for amqssub

notes

```
# Notice that because the test queue manager has AMS, the code is trying to check if the Subscription-Queue
# is managed by AMS (which is not).
# Thus, we are ignoring the MQOPEN, MQGET and MQCLOSE for SYSTEM.PROTECTION.POLICY.QUEUE.
..
 08:01:42.965602    31675.1      CONN:5400006 }  MQSUB rc=OK FunctionTime=5068
 08:01:42.965613    31675.1      CONN:5400006 {  MQGET
 08:01:42.965614    31675.1      CONN:5400006 -{  zstMQGET
.
# Notice that the MQGET is done on the managed queue: SYSTEM.MANAGED.NDURABLE…
 08:01:48.924957    31675.1      CONN:5400006      MQGET <<
08:01:48.924987    31675.1      CONN:5400006      Hobj:
 08:01:48.924993    31675.1      CONN:5400006       0x0000:  04000000                        |....           |
 08:01:48.924999    31675.1      CONN:5400006       ObjHdl:4 ObjType:SUBSCRIPTION-QUEUE
 08:01:48.925005    31675.1      CONN:5400006      Msgdesc:
 08:01:48.925010    31675.1      CONN:5400006       0x0000:  4d442020 02000000 00000000 08000000  |MD  ............|
 08:01:48.925021    31675.1      CONN:5400006       0x0010:  00000000 00000000 53595354 454d2e4d  |........SYSTEM.M|
 08:01:48.925021    31675.1      CONN:5400006       0x0020:  414e4147 45442e4e 44555241 424c452e  |ANAGED.NDURABLE.|
 08:01:48.925021    31675.1      CONN:5400006       0x0030:  35393943 33313839 32303034 37363731  |599C318920047671|
 08:01:48.925021    31675.1      CONN:5400006       0x0040:  20202020 20202020                    |                |
 08:01:48.925026    31675.1      CONN:5400006      Bufferlength:
 08:01:48.925032    31675.1      CONN:5400006       0x0000:  ff030000                        |....           |

# The receiver message text is: TEST-PUB
 08:01:48.925038    31675.1      CONN:5400006      Buffer:
 08:01:48.925044    31675.1      CONN:5400006       0x0000:  54455354 2d505542                |TEST-PUB       |
```

Notes: Activity Trace for amqssub (part 1)

notes

Original file name: QMPS-activity-trace-1.txt

```
MonitoringType: MQI Activity Trace
Correl_id:
00000000:  414D 5143 514D 5053 2020 2020 2020 2020  'AMQCQMPS        '
00000010:  8931 9C59 6F76 0420                      '.1.Yov.         '
QueueManager: 'QMPS'
Host Name: 'mosquito'
IntervalStartDate: '2017-08-23'
IntervalStartTime: '08:01:42'
IntervalEndDate: '2017-08-23'
IntervalEndTime: '08:02:18'
CommandLevel: 903
SeqNumber: 0
ApplicationName: 'amqssub'
Application Type: MQAT_UNIX
ApplicationPid: 31675
UserId: 'mqm'
API Caller Type: MQXACT_EXTERNAL
API Environment: MQXE_OTHER
Appl Function Type: MQFUN_TYPE_UNKNOWN
```

Notes: Activity Trace for amqssub (part 2)

notes

```
Pointer size: 8
Platform: MQPL_UNIX
=====================================================================
  Tid Date       Time       Operation       CompCode       MQRC  HObj
  001 2017-08-23 08:01:42   MQXF_CONN       MQCC_OK        0000  -
  001 2017-08-23 08:01:42   MQXF_SUB        MQCC_OK        0000  4
  001 2017-08-23 08:01:42   MQXF_INQ        MQCC_OK        0000  4
  001 2017-08-23 08:01:42   MQXF_GET        MQCC_OK        0000  4
  001 2017-08-23 08:01:48   MQXF_GET        MQCC_FAILED    2033  4
  001 2017-08-23 08:02:18   MQXF_CLOSE      MQCC_OK        0000  2
  001 2017-08-23 08:02:18   MQXF_CLOSE      MQCC_OK        0000  4
  001 2017-08-23 08:02:18   MQXF_DISC       MQCC_OK        0000  -
=====================================================================
```

Notice:

- The **MQXF_INQ** was done to fine out the SYSTEM.MANAGED.NDURABLE.x queue.
- The 2nd MQGET "failed", after the timeout, there were no further messages (rc 2033 MQRC_NO_MSG_AVAILABLE) and thus, the MQGET terminated.

```
 001 2017-08-23 08:01:48  MQXF_GET        MQCC_FAILED   2033
```

- The first MQXF_CLOSE is for the Subscription, and the second close is for the destination queue.

Notes: Source amqssbxa.c

notes

- Comments from the source code that are relevant in this presentation.

```
/* Description: Sample C program that subscribes and gets messages   */
/*              from a topic (example using MQSUB) allowing the use   */
/*              of extended options on the MQSUB call, over and       */
/*              above those available on the simpler MQSUB sample.    */

/*   AMQSSBXA has the following parameters                            */
/*       required, at least one of:                                   */
/*            -t <string>  Topic string                               */
/*            -o <name>    Topic object name                          */
/*       optional:                                                    */
/*            -m <name>    Queue manager name                         */
/*            -b <type>    Connection binding type (def = standard)   */
/*            -q <name>    Destination queue name                     */
/*            -w <seconds> Wait interval on MQGET (def = 30)          */
/*                          unlimited     MQWI_UNLIMITED             */
/*                          none          no wait                    */
/*                          n             wait interval in seconds   */
/*            -d <subname> Create/resume named durable subscription   */
/*            -k           Keep durable subscription on MQCLOSE       */
```

Notes: Source amqssbxa.c

notes

The code uses similar MQCONN, MQSUB, etc, as amqssuba.c

The bulk of the code is for handling the many input arguments and handling the options.

Notes: Normal trace for amqssbx

notes

Original file name: AMQ31674.0.FMT
Rename file name: AMQ31674. amqssbx.FMT
Command: amqssbx -m QMPS -d SUB20 -q Q3 -t sales –k
(Using durable subscription SUB20 with provided queue Q3 using topic string ‘sales’, allow to resume)

```
08:01:41.271613    31674.1      CONN:5400006 {  MQSUB
# The topic string is: sales
# The durable subscriber name is SUB20 and the provided queue is Q3
 08:01:41.271656    31674.1      CONN:5400006      ObjectString:
 08:01:41.271657    31674.1      CONN:5400006        0x0000:  73616c65 73                |sales          |
 08:01:41.271658    31674.1      CONN:5400006      SubName:
 08:01:41.271659    31674.1      CONN:5400006        0x0000:  53554232 30                |SUB20          |
 08:01:41.271660    31674.1      CONN:5400006      Hobj:
 08:01:41.271661    31674.1      CONN:5400006        0x0000:  02000000                   |....           |
 08:01:41.271662    31674.1      CONN:5400006        ObjHdl:2 ObjType:QUEUE ObjName:Q3 ResObjName:Q3

# This sample can also handle message properties, which use: MQCRTMH (it does not apply for this scenario)
 08:01:41.273350    31674.1      CONN:5400006      MQCRTMH >>

# The message payload is: TEST-PUB
08:01:48.929455    31674.1      CONN:5400006      MQGET <<
 08:01:48.929525    31674.1      CONN:5400006      Buffer:
 08:01:48.929529    31674.1      CONN:5400006        0x0000:  54455354 2d505542          |TEST-PUB       |
```

Notes: Activity Trace for amqssbx (part 1)

notes

Original file name: QMPS-activity-trace-1.txt

MonitoringType: MQI Activity Trace
Correl_id:
00000000: 414D 5143 514D 5053 2020 2020 2020 2020 'AMQCQMPS   '
00000010: 8931 9C59 AECF 0420   '.1.Y...   '
QueueManager: 'QMPS'
Host Name: 'mosquito'
IntervalStartDate: '2017-08-23'
IntervalStartTime: '08:01:41'
IntervalEndDate: '2017-08-23'
IntervalEndTime: '08:02:18'
CommandLevel: 903
SeqNumber: 0
ApplicationName: 'amqssbx'
Application Type: MQAT_UNIX
ApplicationPid: 31674
UserId: 'mqm'
API Caller Type: MQXACT_EXTERNAL
API Environment: MQXE_OTHER

Notes: Activity Trace for amqssbx (part 2)

notes

Application Function: ''
Appl Function Type: MQFUN_TYPE_UNKNOWN
Trace Detail Level: 3
Trace Data Length: 300
Pointer size: 8
Platform: MQPL_UNIX

```
==========================================================================
  Tid Date       Time       Operation       CompCode       MQRC  HObj
  001 2017-08-23 08:01:41   MQXF_CONNX      MQCC_OK        0000  -
  001 2017-08-23 08:01:41   MQXF_OPEN       MQCC_OK        0000  2 (Q3)
  001 2017-08-23 08:01:41   MQXF_SUB        MQCC_OK        0000  2 (Q3)
  001 2017-08-23 08:01:41   MQXF_GET        MQCC_OK        0000  2 (Q3)
  001 2017-08-23 08:01:48   MQXF_GET        MQCC_FAILED    2033  2 (Q3)
  001 2017-08-23 08:02:18   MQXF_CLOSE      MQCC_OK        0000  4
  001 2017-08-23 08:02:18   MQXF_CLOSE      MQCC_OK        0000  2 (Q3)
  001 2017-08-23 08:02:18   MQXF_DISC       MQCC_OK        0000  -
==========================================================================
```

Notice that the highlighted item shows that the MQGET was OK from queue Q3.

The End

This is the end of the presentation.

THANKS!!