MQ Pub/Sub: durable subscribers

http://www.ibm.com/support/docview.wss?uid=swg27050181

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

Agenda

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

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

Durable Subscribers – administrative objects that survive a restart of the queue manager
Also called “Unmanaged subscribers”
In contrast, **non-durable** are also called “**managed**”.

Scenario 1

- You want to have a Subscriber that receives published messages for a topic, but if the queue manager is restarted, and the subscriber is reconnected to the queue manager, the subscriber should continue to receive published messages for the topic.

Example:

- Publisher publishes 3 messages.
- SUB1 receives 3 messages and consumes them.
- Queue manager restarts.
- Publisher resumes publishing 2 more messages.
- SUB1 resumes and receives 2 more messages.

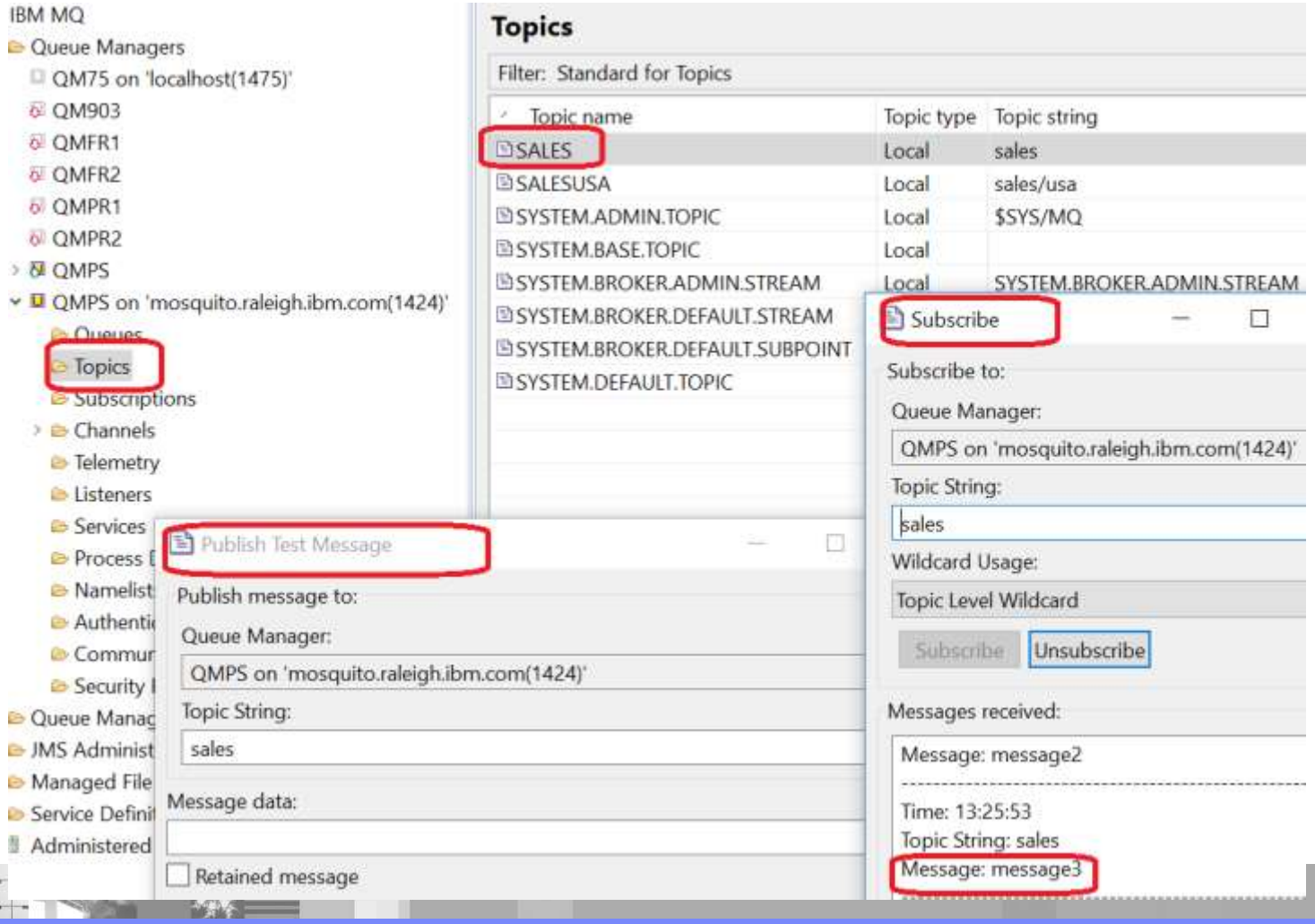
Scenario 2

- Similar to Scenario 1, but the Subscriber application ends, but remains subscribed, and when it reconnects and resumes, the Subscriber receives the messages published during the app outage.
- Example:
- Publisher publishes 10 messages.
- SUB1 receives 10 messages.
- SUB1 stops.
- Queue manager continues working.
- Publisher publishes 3 more messages.
- SUB1 resumes and receives 3 more messages.

Scenario 1 – using non-durable (pitfall)

- Using
- Non-durable
- Subscription
- for ‘sales’

- Subscriber
- receives
- 3 messages



Scenario 1 – tpstatus, type(sub)

- **display tpstatus('sales') type(sub)**
- AMQ8754I: Display topic status details.
- TOPICSTR(sales)
- **SUBID(414D5120514D5053202020202020202089319C596CEA0320)**
- SUBUSER(rivera) RESMDATE(2017-08-22)
- RESMTIME(09:29:13) LMSGDATE(2017-08-22)
- LMSGTIME(09:29:27)
- ACTCONN(414D5143514D5053202020202020202089319C5969EA0320)
- **DURABLE(NO)** **SUBTYPE(API)**
- MCASTREL(,) **NUMMSGS(3)**

- Notice these attributes:
- SUBID(414D5120514D5053202020202020202089319C59**6CEA0320**)
- DURABLE(NO) => Non durable
- SUBTYPE(API) => Created by application (API)
- NUMMSGS(3) => Number of messages received: 3

Scenario 1 – subid, temp dynamic queue

- **display sub SUBID(414D5120514D50532020202020202020202089319C596CEA0320)**
- AMQ8096I: IBM MQ subscription inquired.
- SUBID(414D5120514D50532020202020202020202089319C596CEA0320)
- SUB( ) TOPICSTR(sales)
- TOPICOBJ( ) DISTYPE(RESOLVED)
- **DEST(SYSTEM.MANAGED.NDURABLE.599C31892003EA6B)**
- DESTQMGR(QMPS) PUBAPPID( )
- SELECTOR( ) SELTYPE(NONE)
- USERDATA( )
- PUBACCT(000000000000000000000000000000000000000000000000000000000000000000000000)
- DESTCORL(414D5120514D50532020202020202020202089319C596CEA0320)
- **DESTCLAS(MANAGED) DURABLE(NO)**
- EXPIRY(UNLIMITED) PSPROP(MSGPROP)
- PUBPRTY(ASPUB) REQONLY(NO)
- SUBSCOPE(ALL) SUBLEVEL(1)
- **SUBTYPE(API)** VARUSER(FIXED)
- WSCHEMA(TOPIC) SUBUSER(rivera)
-

- Notice the destination queue, which is a temporary dynamic queue.
- DEST(SYSTEM.MANAGED.NDURABLE.599C31892003**EA6B**)
- DESTCLAS(MANAGED)

Restart queue manager or application

- Either the queue manager is stopped and restarted, or the application is stopped and restarted.
- Dialogs Test Publication and Subscription need to be redone.
- The Test Publication publishes 1 more message and it is received by the Test Subscription.
- Apparently all is fine…

Restart queue manager or application

- … BUT current subscriber is NOT the same as the previous one! Different SUBID, NUMMSGS, DEST queue.

- NEW: **display tpstatus('sales') type(sub)**
  - SUBID(414D5120514D50532020202020202089319C**5973EA0320**)
  - NUMMSGS(**1**)
- OLD data:
  - SUBID(414D5120514D50532020202020202089319C**596CEA0320**)
  - NUMMSGS(**3**)

- NEW: **display sub SUBID(414D5120514D50532020202020202089319C5973EA0320) dest**
  - SUB( )
  - DEST(SYSTEM.MANAGED.NDURABLE.599C31892003EA**72**)
- OLD:
  - DEST(SYSTEM.MANAGED.NDURABLE.599C31892003EA**6B**)

Notes: Creating durable subscription

notes

- To address the scenarios, it is necessary to create a DURABLE subscription. Also called an “unmanaged Subscriber”.

- For the destination queue, there are 2 options:

- 1) “Managed” => the queue manager will create a permanent dynamic queue, and manage it. It will be deleted by the qmgr when the subscription is deleted.

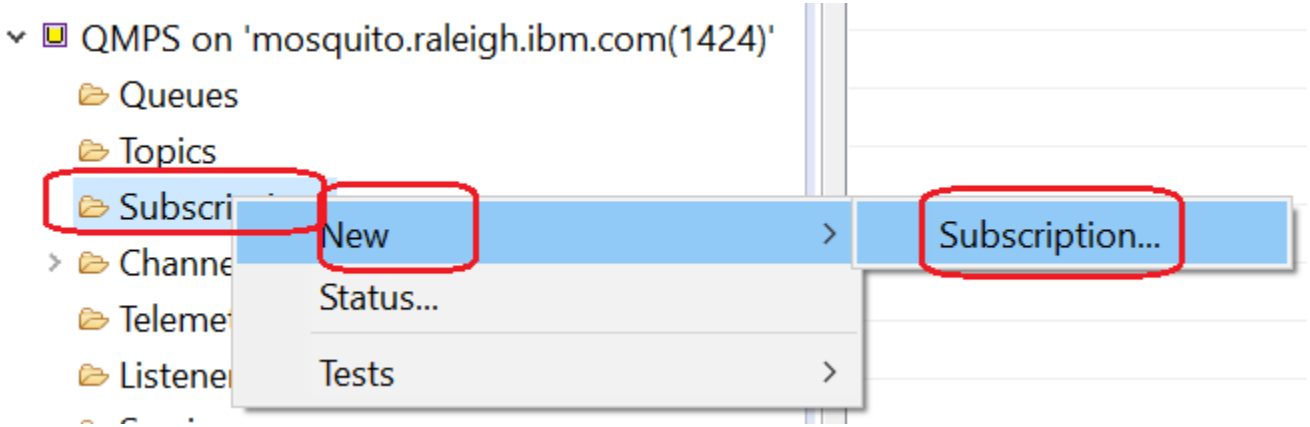
- 2) “Provided” => User defines the queue. This queue is NOT deleted by the queue manager when the subscription is deleted.

- Reference:

This link explains the distinction between managed and unmanaged subscriptions:
https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_8.0.0/com.ibm.mq.dev.doc/q026620_.htm

Creating durable subscription - managed

- Using MQ Explorer.
- Select folder “Subscriptions”, right click “New” then
- “Subscription…”

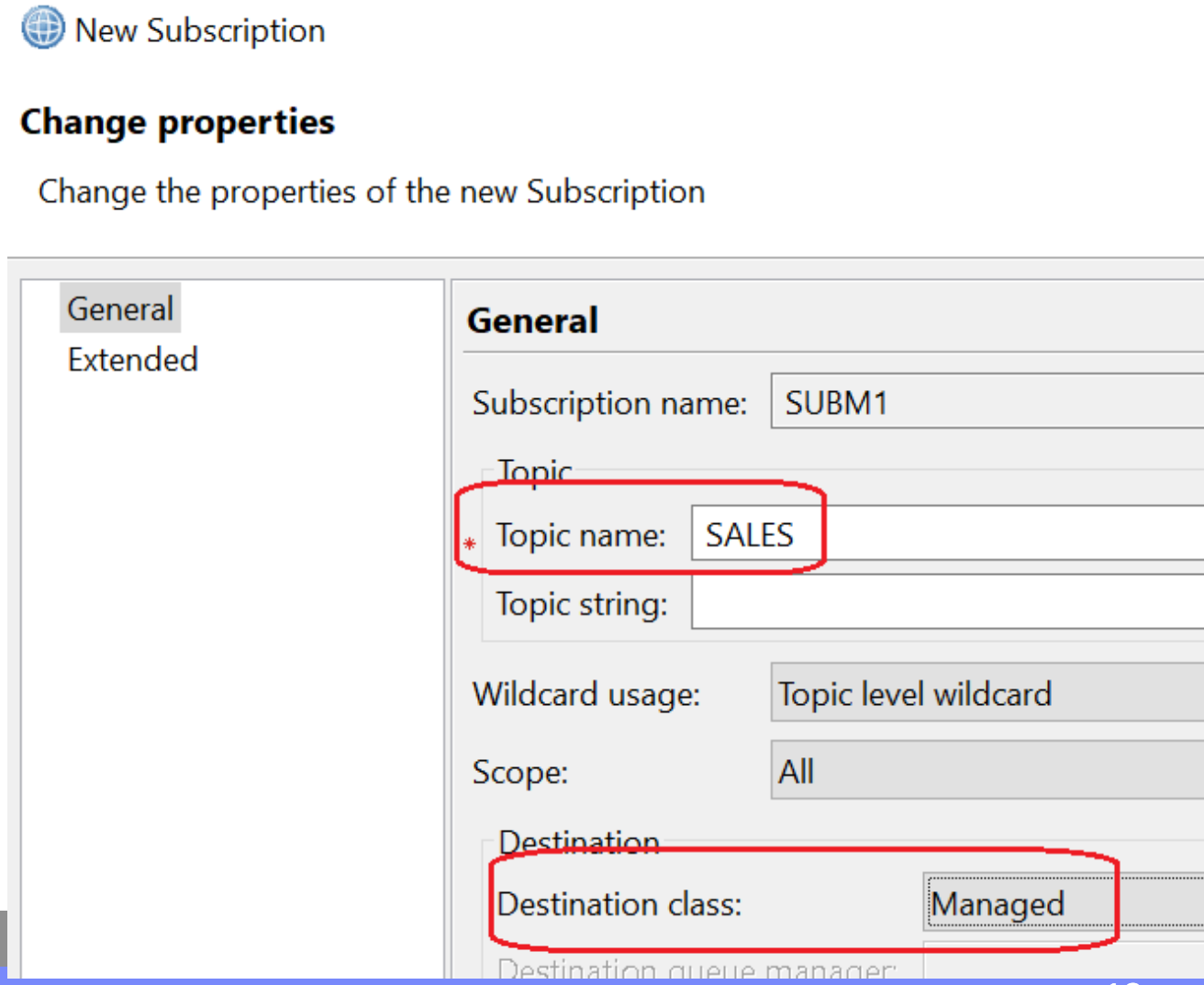


Creating durable subscription - managed

- Let's call it:
- SUBM1

- Topic name:
- (OBJECT!!)
- SALES

- Destination
- class:
  Managed



Creating durable subscription, runmqsc

define sub('SUBM2') topicobj(SALES) destclas(MANAGED)
AMQ8094I: IBM MQ subscription created.

display SUB('SUBM2')
AMQ8096I: IBM MQ subscription inquired.
SUBID(414D5120514D50532020202020202089319C5970EC0320)
SUB(SUBM2) **TOPICSTR(sales)**
TOPICOBJ(SALES) DISTYPE(RESOLVED)
DEST(SYSTEM.MANAGED.DURABLE.599C31892003EC6F)
DESTQMGR(QMPS) PUBAPPID( )
SELECTOR( ) SELTYPE(NONE)
USERDATA( )
PUBACCT(043130323000000000000000000000000000000000000000000000000000000006)
DESTCORL(414D5120514D50532020202020202089319C5970EC0320)
DESTCLAS(MANAGED) **DURABLE(YES)**
EXPIRY(UNLIMITED) PSPROP(MSGPROP)
PUBPRTY(ASPUB) REQONLY(NO)
SUBSCOPE(ALL) SUBLEVEL(1)
SUBTYPE(ADMIN) VARUSER(ANY)
WSCHEMA(TOPIC) SUBUSER(rivera)

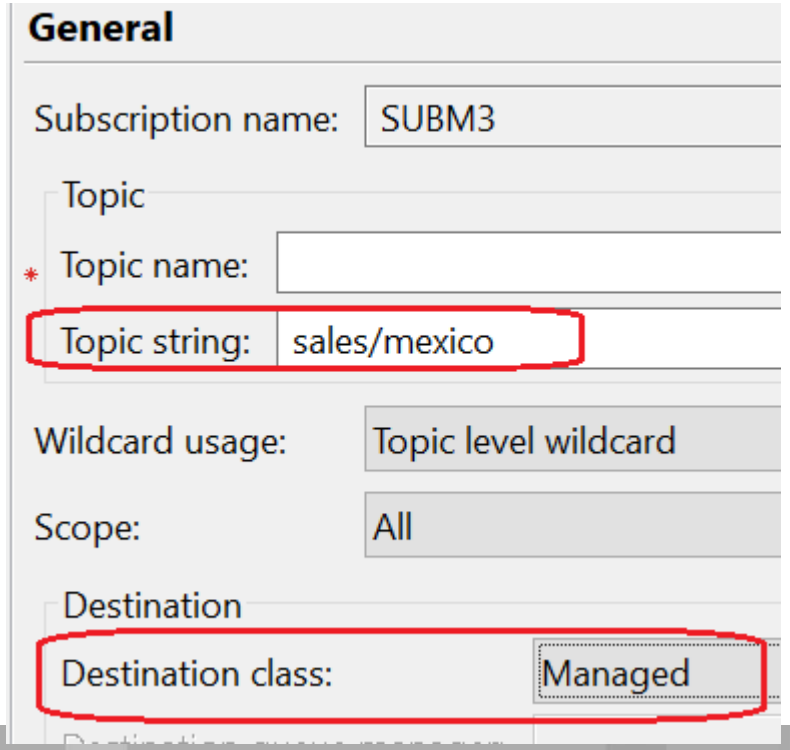
Creating durable subscription - managed

- Using topic string that does not have a Topic Object

- Let's call it:
- SUBM3

- Topic string:
- sales/mexico

- Destination class:
  Managed



Creating durable subscription, runmqsc

- Example of using topic string even when there is a Topic Object

- **define sub('SUBM4') topicstr('sales/usa') destclas(MANAGED)**

AMQ8094I: IBM MQ subscription created.

- **display sub('SUBM4')**

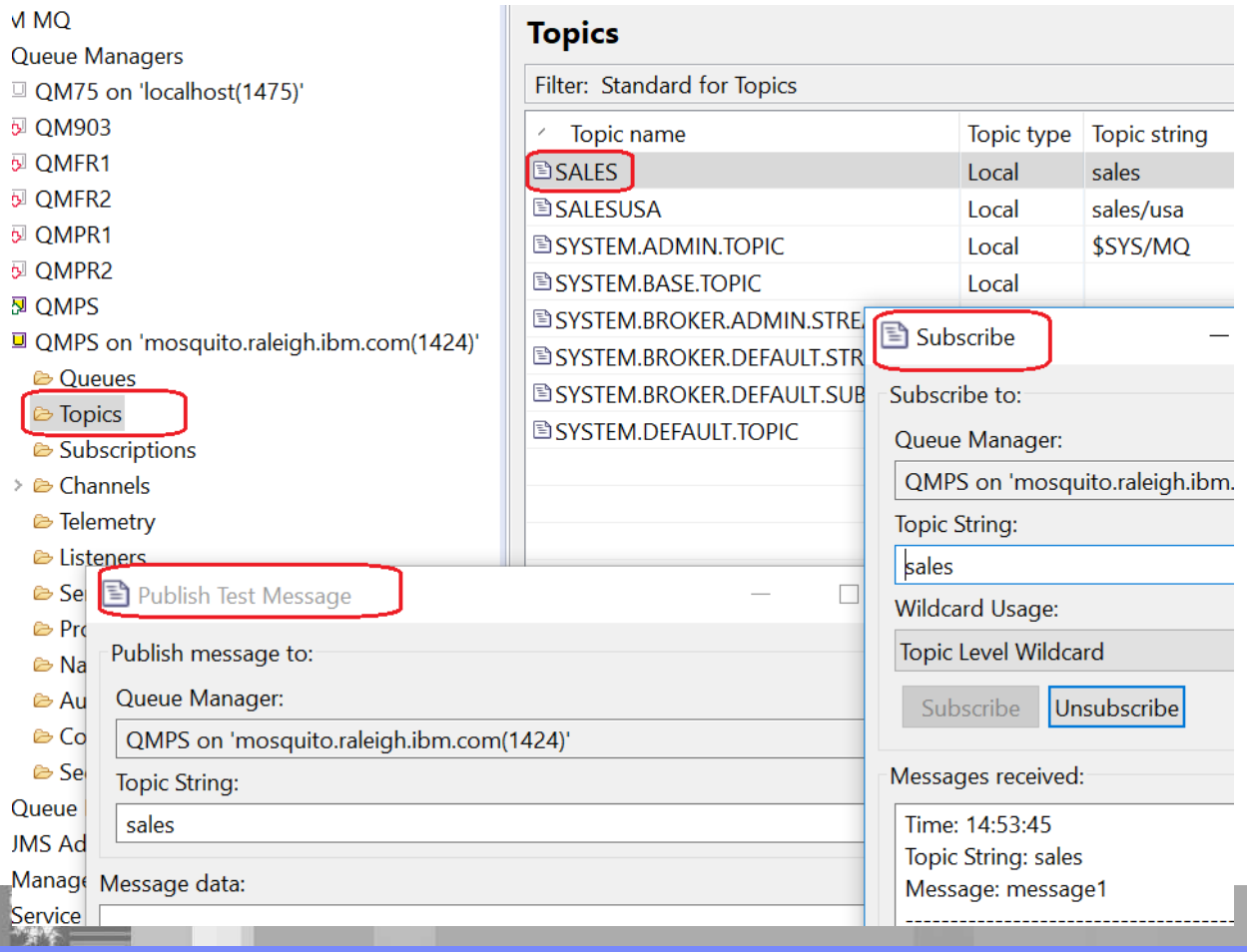
AMQ8096I: IBM MQ subscription inquired.
SUBID(414D5120514D50532020202020202089319C5976EC0320)
SUB(SUBM4) TOPICSTR(sales/usa)
TOPICOBJ( ) DISTYPE(RESOLVED)
DEST(SYSTEM.MANAGED.DURABLE.599C31892003EC75)
DESTQMGR(QMPS) PUBAPPID( )
SELECTOR( ) SELTYPE(NONE)
USERDATA( )
PUBACCT(043130323000000000000000000000000000000000000000000000000000000006)
DESTCORL(414D5120514D50532020202020202089319C5976EC0320)
DESTCLAS(MANAGED) DURABLE(YES)
EXPIRY(UNLIMITED) PSPROP(MSGPROP)
PUBPRTY(ASPUB) REQONLY(NO)
SUBSCOPE(ALL) SUBLEVEL(1)
SUBTYPE(ADMIN) VARUSER(ANY)
WSCHEMA(TOPIC) SUBUSER(rivera)

Open test dialogs

- Let's open dialogs
- for:
- Test Publication
- Test Subscription

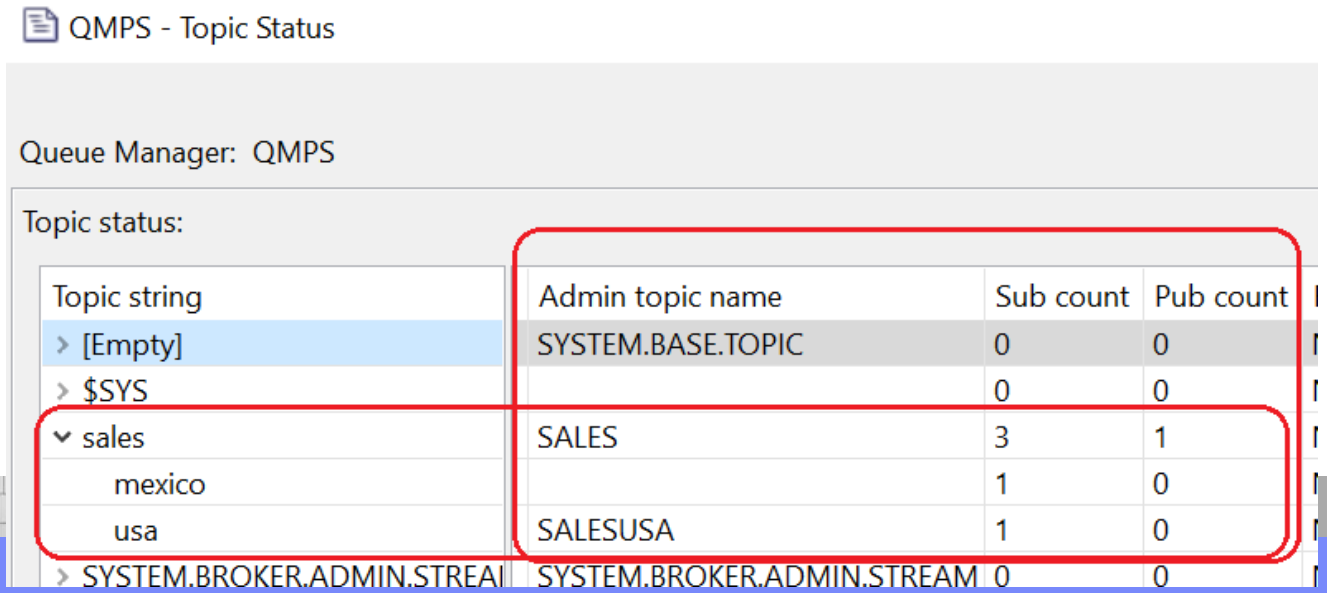
- The reason is to
- have a non-durable
- Subscription for
- Comparison.

- Publish 1 message.



Topic Status window

- See Topic Status window and expand items.
- ‘sales’ => 1 test publication dialog
  => 3 subscribers: 1 test sub dialog (non-durable)
  => 2 durable: SUBM1 and SUBM2
- ‘sales/mexico’ => 1 durable: SUBM3 (no Topic Object)
- ‘sales/usa’ => 1 durable: SUBM4 (topic object SALESUSA)

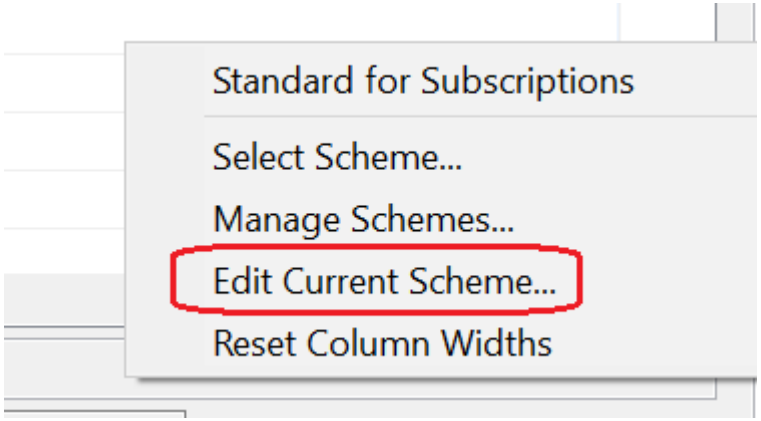


Subscriptions window

Scheme for the Subscriptions was modified to rearrange the columns.
From “Scheme Standard for Subscriptions – Distributed”, hove the cursor on the mark on the right side of the bar…

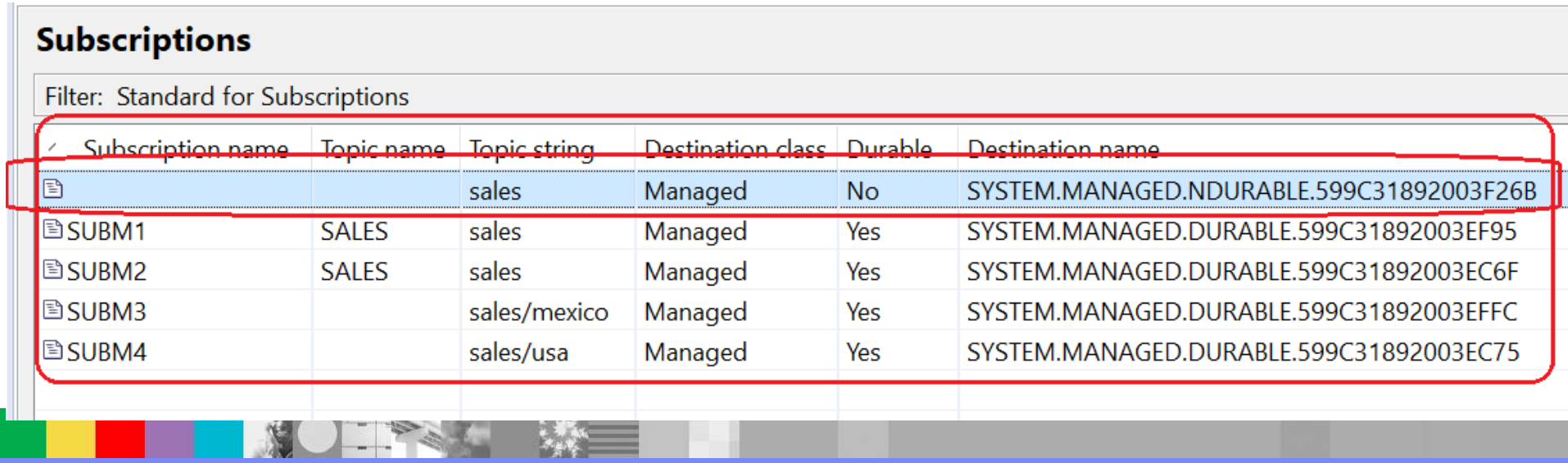
Subscriptions window

- From the context dialog, select
- “Edit Current Scheme”
- You will see a dialog where you can arrange the order of the rows.



Subscriptions window – non durable

- Notice that the top row is for the ‘Test Subscription’ dialog, which is non-durable and has an associated temporary dynamic queue whose name is of the format:
  - SYSTEM.**MANAGED.NDURABLE**….



Subscriptions

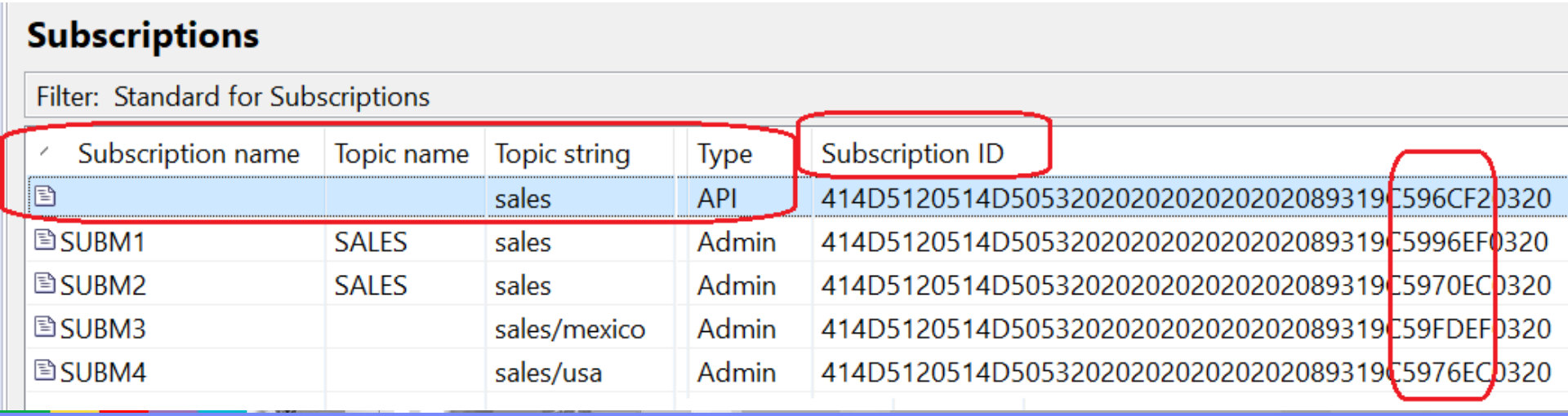
Filter: Standard for Subscriptions

Subscription name	Topic name	Topic string	Destination class	Durable	Destination name
		sales	Managed	No	SYSTEM.MANAGED.NDURABLE.599C31892003F26B
SUBM1	SALES	sales	Managed	Yes	SYSTEM.MANAGED.DURABLE.599C31892003EF95
SUBM2	SALES	sales	Managed	Yes	SYSTEM.MANAGED.DURABLE.599C31892003EC6F
SUBM3		sales/mexico	Managed	Yes	SYSTEM.MANAGED.DURABLE.599C31892003EFFC
SUBM4		sales/usa	Managed	Yes	SYSTEM.MANAGED.DURABLE.599C31892003EC75

Subscriptions window – non durable

- Notice that the non-durable has a Type of “API”, while the durable ones have a Type of “Admin”.

- Notice that the Subscription ID is different for each subscription.



Subscriptions – managed queues

Non-durable subscriptions use the model queue:

SYSTEM.**NDURABLE**.MODEL.QUEUE

Example of **temporary** dynamic queue:

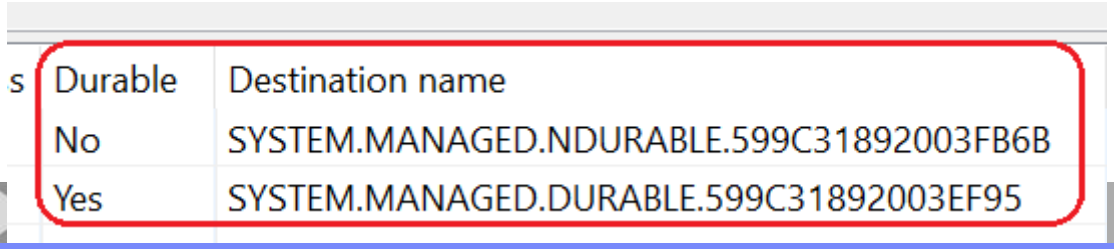
SYSTEM.MANAGED.NDURABLE.599C31892003FB6B

Durable subscriptions use the model queue:

SYSTEM.**DURABLE**.MODEL.QUEUE

Example of **permanent** dynamic queue:

- SYSTEM.MANAGED.DURABLE.599C31892003EF95



s	Durable	Destination name
	No	SYSTEM.MANAGED.NDURABLE.599C31892003FB6B
	Yes	SYSTEM.MANAGED.DURABLE.599C31892003EF95

Notes: Managed dynamic queues

notes

Using “display ql” (or “display queue”) to show

** Non-durable: temporary dynamic

display ql(SYSTEM.MANAGED.**NDURABLE**.599C31892003FB6B) deftype descr
QUEUE(SYSTEM.MANAGED.NDURABLE.599C31892003FB6B)
TYPE(QLOCAL) DEFTYPE(**TEMPDYN**)
DESCR(Model for managed queues for **non durable** subscriptions)

** Durable: permanent dynamic

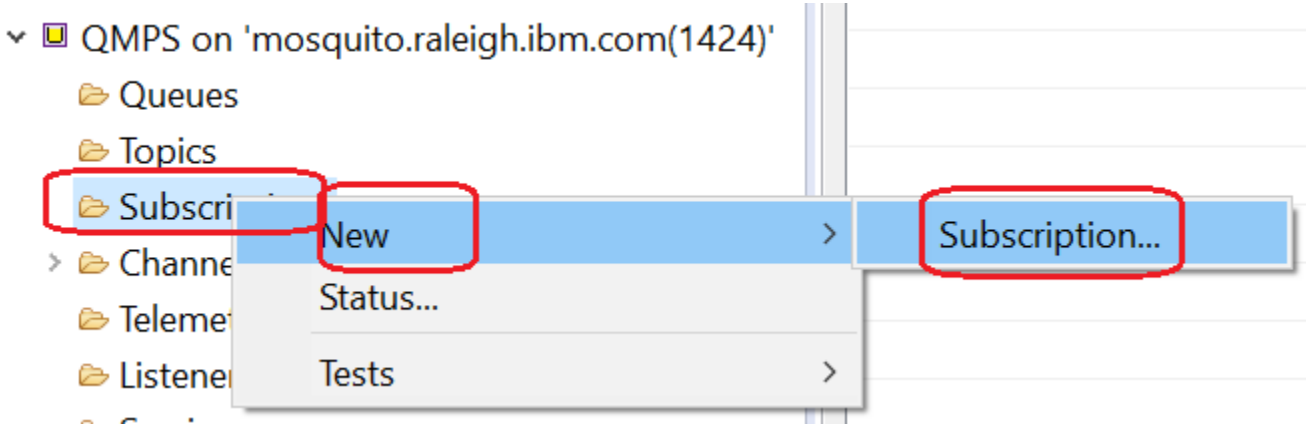
display ql(SYSTEM.MANAGED.**DURABLE**.599C31892003EF95) deftype descr
QUEUE(SYSTEM.MANAGED.DURABLE.599C31892003EF95)
TYPE(QLOCAL) DEFTYPE(**PERMDYN**)
DESCR(Model for managed queues for **durable** subscriptions)

Notice that “DESCR” (Description) is inherited from the model queue.
This field is a clue about which is the Model queue used to generate a dynamic queue.

Creating durable subscription - provided

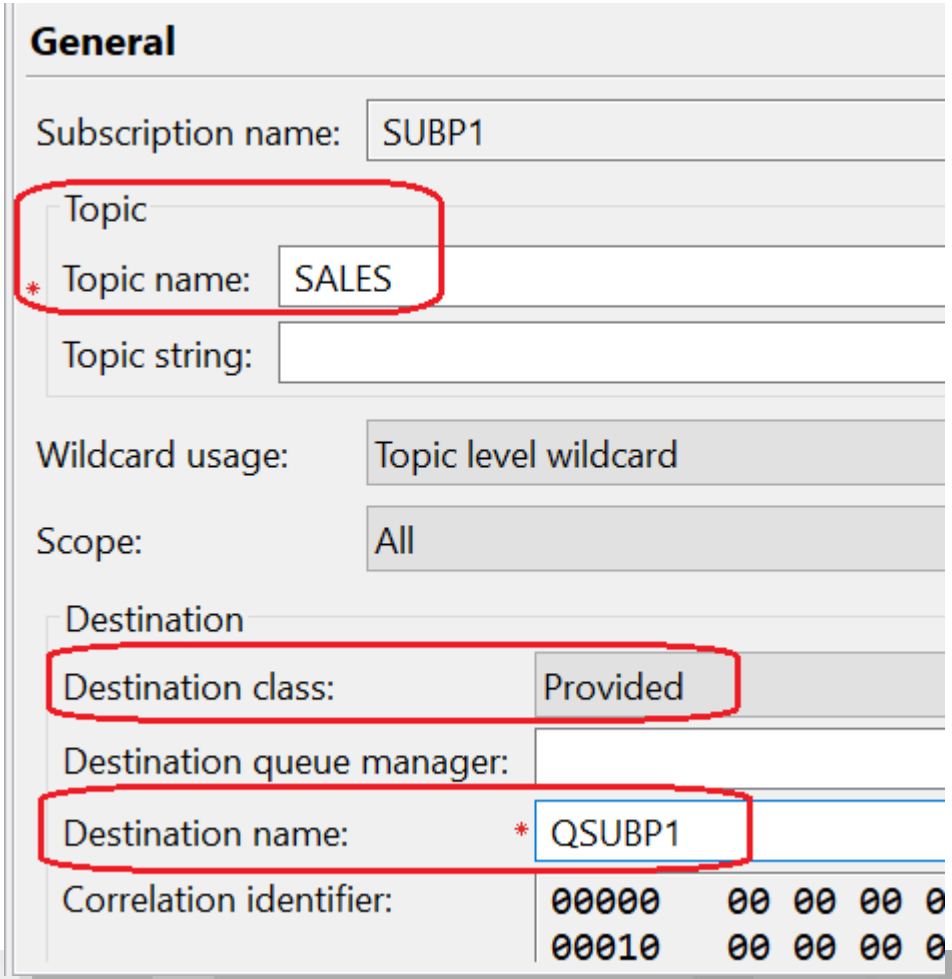
- 1: Create queue such as: QSUBP1

- 2: Then, from MQ Explorer, Select folder “Subscriptions”, right click “New” then “Subscription…”



Creating durable subscription - provided

- Let's call it:
  - SUBP1
- Topic name: (OBJECT!!)
  - SALES
- Destination class:
  Provided
- Destination name (queue):
  - QSUBP1



Creating durable subscription - provided

- Let's create another subscription.
- *** NOTICE ***
- The destination queue has a typo (on purpose)
- A later slide will show you how to handle and fix it.

- Let's call it: SUBP2
- Topic name: SALES
- Destination class: Provided
- Destination name (queue): QSUBP1
- Note: It should be QSUBP2 !! (to be fixed later!!)

Subscription window

- The updated list of Subscriptions is shown below.

- The SUBP1 sub uses a Provided queue: QSUBP1
- The SUBP2 sub uses a Provided queue:
- QSUBP1 (later on will be changed to QSUBP2)

Subscription name	Topic name	Topic string	Destination class	Durable	Destination name	Type
		sales	Managed	No	SYSTEM.MANAGED.NDURABLE.599C31892003FB6B	API
SUBM1	SALES	sales	Managed	Yes	SYSTEM.MANAGED.DURABLE.599C31892003EF95	Admin
SUBM2	SALES	sales	Managed	Yes	SYSTEM.MANAGED.DURABLE.599C31892003EC6F	Admin
SUBM3		sales/mexico	Managed	Yes	SYSTEM.MANAGED.DURABLE.599C31892003EFFC	Admin
SUBM4		sales/usa	Managed	Yes	SYSTEM.MANAGED.DURABLE.599C31892003EC75	Admin
SUBP1	SALES	sales	Provided	Yes	QSUBP1	Admin
SUBP2	SALES	sales	Provided	Yes	QSUBP1	Admin

Queues window

- The following composite view shows the destination queues for our subscriptions.
- The “Current queue depth” (CURDEPTH) is 0.

Filter: Standard for Queues

Queue name	Queue type	Open input count	Open output count	Current queue depth
AMQ.MQEXPLORER.599C31892003F96B	Local	1	1	0
Q1	Local	0	0	0
QSUBP1	Local	0	0	0
QSUBP2	Local	0	0	0
SYSTEM.JMS.TEMPQ.MODEL	Model			
SYSTEM.MANAGED.DURABLE.599C31892003EC6F	Local	0	0	0
SYSTEM.MANAGED.DURABLE.599C31892003EC75	Local	0	0	0
SYSTEM.MANAGED.DURABLE.599C31892003EF95	Local	0	0	0
SYSTEM.MANAGED.DURABLE.599C31892003EFFC	Local	0	0	0
SYSTEM.MANAGED.NDURABLE.599C31892003FB6B	Local	1	0	0

Queues window

- Publish message: TEST-PUB-MSG
- Refresh view of the queues.
- The “Current queue depth” (CURDEPTH) was updated for some queues.

QSUBP1	Local	0	0	2
QSUBP2	Local	0	0	0
SYSTEM.JMS.TEMPQ.MODEL	Model			
SYSTEM.MANAGED.DURABLE.599C31892003EC6F	Local	0	0	1
SYSTEM.MANAGED.DURABLE.599C31892003EC75	Local	0	0	0
SYSTEM.MANAGED.DURABLE.599C31892003EF95	Local	0	0	1
SYSTEM.MANAGED.DURABLE.599C31892003EFFC	Local	0	0	0
SYSTEM.MANAGED.NDURABLE.599C31892003FB6B	Local	1	0	0

Notes: Queues

notes

- Notice that the following queues received messages:

- 2 messages (1 extra due to a typo for SUBP2): QSUBP1

1 message: SYSTEM.MANAGED.DURABLE.599C31892003EC6F
SYSTEM.MANAGED.DURABLE.599C31892003EF95

Question: Given a queue, how to find the corresponding durable sub?
Answer: You can use "runmqsc" with the following query.
Using queue QSUBP1 as a concrete example:
display sub(*) where(dest eq 'QSUBP1')
.
Notice that for QSUBP1 there are 2 entries:

AMQ8096I: IBM MQ subscription inquired.
SUBID(414D5120514D50532020202020202089319C59E3070420)
SUB(SUBP1) DEST(QSUBP1)

AMQ8096I: IBM MQ subscription inquired.
SUBID(414D5120514D50532020202020202089319C59E7080420)
SUB(SUBP2) DEST(QSUBP1)

Notes: Queues

notes

- Here are the subscriptions associated with the dynamic queues:

- display sub(*) where(dest eq 'SYSTEM.MANAGED.DURABLE.599C31892003EC6F')
  - SUBID(414D5120514D50532020202020202089319C5970EC0320)
  - **SUB(SUBM2)**
  - DEST(SYSTEM.MANAGED.DURABLE.599C31892003EC6F)

- display sub(*) where(dest eq 'SYSTEM.MANAGED.DURABLE.599C31892003EF95')
  - SUBID(414D5120514D50532020202020202089319C5996EF0320)
  - **SUB(SUBM1)**
  - DEST(SYSTEM.MANAGED.DURABLE.599C31892003EF95)

Notes: Finding topic strings for subs

notes

- We now know that the following subscriptions received messages:
- SUBP1 SUBP2 SUBM1 SUBM2
- Question: What is the topic string associated with them?
- Answer: **display sub(SUB*) topicstr**
- It is: TOPICSTR(sales)

- AMQ8096I: IBM MQ subscription inquired.
  - SUBID(414D5120514D505320202020202020089319C5996EF0320)
  - **SUB(SUBM1) TOPICSTR(sales)**
- AMQ8096I: IBM MQ subscription inquired.
  - SUBID(414D5120514D505320202020202020089319C59FDEF0320)
  - SUB(SUBM3) TOPICSTR(sales/mexico)
- AMQ8096I: IBM MQ subscription inquired.
  - SUBID(414D5120514D505320202020202020089319C5970EC0320)
  - **SUB(SUBM2) TOPICSTR(sales)**
- AMQ8096I: IBM MQ subscription inquired.
  - SUBID(414D5120514D505320202020202020089319C59E3070420)
  - **SUB(SUBP1) TOPICSTR(sales)**
- AMQ8096I: IBM MQ subscription inquired.
  - SUBID(414D5120514D505320202020202020089319C59E7080420)
  - **SUB(SUBP2) TOPICSTR(sales)**

- Note if you know the name of the topic string, you could use:
  - **display sub(SUB*) where(topicstr eq 'sales')**

Notes: Each received message is unique

notes

- Let's use the MQ sample "amqsbcg" to browse (non destructive get) the contents of the destination queues.
- For example: **amqsbcg QSUBP1 QMPS**

- The queue QSUBP1 has 2 messages.
- Let's take a closer look at the 1st.
- Only some attributes will be shown from the Message descriptor.
- But the whole "payload" (contents) is shown.

```
MQGET of message number 1, CompCode:0 Reason:0
****Message descriptor****
StrucId  : 'MD  '  Version : 2
Format : 'MQSTR   '
MsgId :    X'414D5120514D50532020202020202089319C59750A0420'
CorrelId : X'414D5120514D50532020202020202089319C59E7080420'
PutApplName    : 'QMPS                          '
PutDate  : '20170823'    PutTime  : '04241200'
****   Message      ****
length - 12 of 12 bytes
00000000:  5445 5354 2D50 5542 2D4D 5347              'TEST-PUB-MSG    '
```

Notes: Each received message is unique

notes

- Let's take a closer look at the 2nd

```
MQGET of message number 1, CompCode:0 Reason:0
****Message descriptor****
StrucId  : 'MD  '  Version : 2
Format : 'MQSTR   '
MsgId :    X'414D5120514D50532020202020202089319C59760A0420'
CorrelId : X'414D5120514D50532020202020202089319C59E3070420'
PutApplName    : 'QMPS                          '
PutDate  : '20170823'    PutTime  : '04241200'
****   Message      ****
length - 12 of 12 bytes
00000000:  5445 5354 2D50 5542 2D4D 5347              'TEST-PUB-MSG    '
```

- ** Comparison:
- Both messages have the same payload.
- Both messages have ALMOST identical message descriptors…
- - Except for the MsgId (each message is unique!)

```
MsgId :    X'414D5120514D50532020202020202089319C59750A0420'
MsgId :    X'414D5120514D50532020202020202089319C59760A0420'
```

Notes: Each received message is unique

notes

- Let's view the details for the messages on the other destination queues:

- amqsbcg SYSTEM.MANAGED.DURABLE.599C31892003EC6F QMPS

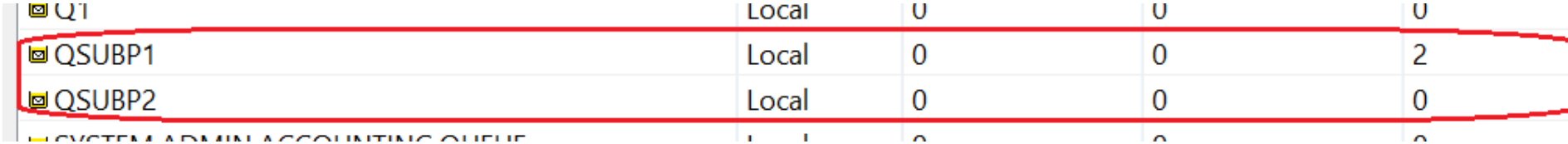
```
  MsgId :    X'414D5120514D50532020202020202089319C5978 0A0420'
  CorrelId : X'414D5120514D50532020202020202089319C5970EC0320'
****   Message      ****
 length - 12 of 12 bytes
00000000:  5445 5354 2D50 5542 2D4D 5347                'TEST-PUB-MSG    '
```

amqsbcg SYSTEM.MANAGED.DURABLE.599C31892003EF95 QMPS

```
  MsgId :    X'414D5120514D50532020202020202089319C5979 0A0420'
  CorrelId : X'414D5120514D50532020202020202089319C5996EF0320'
****   Message      ****
 length - 12 of 12 bytes
00000000:  5445 5354 2D50 5542 2D4D 5347                'TEST-PUB-MSG    '
```

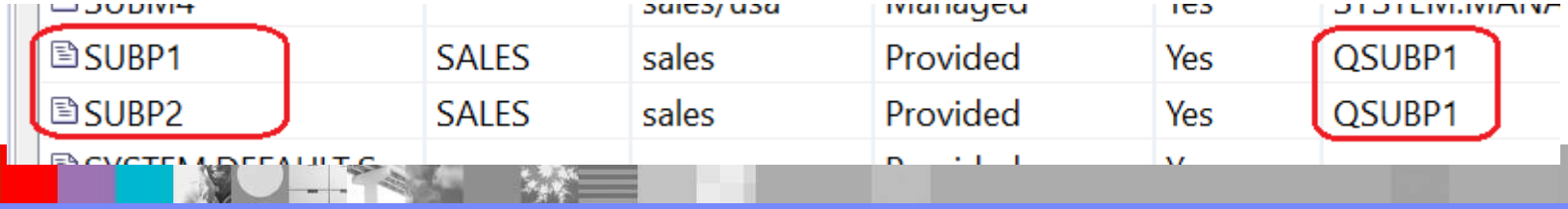
Queues received messages for subs

- Let's return to the issue that the destination queue QSUBP1 has 2 messages
…but QSUBP2 has 0 messages



Q1	Local	0	0	0
QSUBP1	Local	0	0	2
QSUBP2	Local	0	0	0

- QSUBP1 received 1 message for each of the subscriptions that use QSUBP1 as dest queue:
SUBP1 and SUBP2

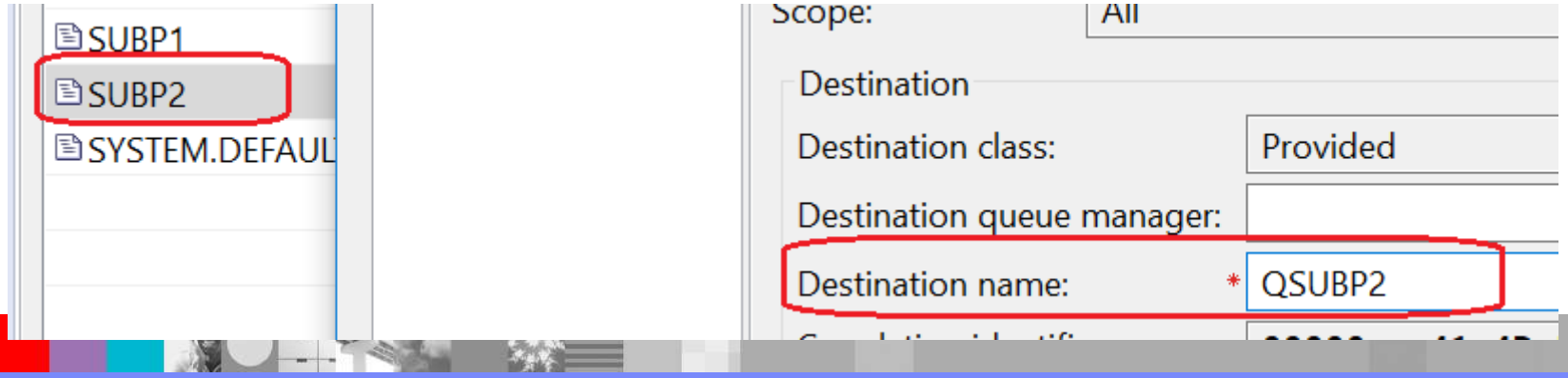


SUBP1	SALES	sales	Provided	Yes	QSUBP1
SUBP2	SALES	sales	Provided	Yes	QSUBP1

Correcting the mistake – ALTER SUB

- Let's correct the mistake.
- From runmqsc:
- ALTER SUB(SUBP2) DEST(QSUBP2)

- From MQ Explorer, bring the Properties for the Sub and modify the Destination Queue:



Correcting the mistake – ALTER SUB

- OK, now we have the proper matching:

subscriber SUBP1 => queue QSUBP1
subscriber SUBP2 => queue QSUBP2

What about the received messages in QSUBP1?
- 1 message is ok, because it was for SUBP1
- But the other message was destined to QSUBP2.

QUESTION: Is there a way to identify that message and move it to the proper destination queue?

Notes: Subid and DestCorl (for Subs)

notes

- QUESTION: Is there a way to identify that message and move it to the proper destination queue?

- Answer: Yes!
- Each subscription has an attribute called the Destination Correlation ID:
- DESTCORL
- By default, the value for DESTCORL is the same as SUBID.

- **display sub(SUBP*) destcorl subid**

```
AMQ8096I: IBM MQ subscription inquired.
     SUBID(414D5120514D50532020202020202089319C59E3070420)
   SUB(SUBP1)
   DESTCORL(414D5120514D50532020202020202089319C59E3070420)

AMQ8096I: IBM MQ subscription inquired.
     SUBID(414D5120514D50532020202020202089319C59E7080420)
   SUB(SUBP2)
   DESTCORL(414D5120514D50532020202020202089319C59E7080420)
```

- Thus:

```
SUB(SUBP1) > DESTCORL(414D5120514D50532020202020202089319C59E3070420)
SUB(SUBP2) > DESTCORL(414D5120514D50532020202020202089319C59E7080420)
```

Notes: DESTCORL > CorrelId in message

notes

- Here is the relationship of Sub and DESTCORL
- SUB(SUBP1) > DESTCORL(414D5120514D5053202020202020202089319C59**E3070420**)
- SUB(SUBP2) > DESTCORL(414D5120514D5053202020202020202089319C59**E7080420**)

The DESTCORL of the subscription is copied into the field "CorrelId" of each message.

The following are the pairs of the fields MsgId and CorrelId for the 2 messages in the queue QSUBP1:

```
MsgId :     X'414D5120514D5053202020202020202089319C59760A0420'
CorrelId : X'414D5120514D5053202020202020202089319C59E3070420'
Note: This message is for sub SUBP1

MsgId :     X'414D5120514D5053202020202020202089319C59750A0420'
CorrelId : X'414D5120514D5053202020202020202089319C59E7080420`
Note: This message is for sub SUBP2
```

Notes: SupportPac MA01 – q.exe

notes

- OK, we know now which message needs to be moved from queue QSUBP1 to QSUBP2.

- Question: How do we move this message?

- Answer: By using the following SupportPac:

- http://www-01.ibm.com/support/docview.wss?rs=171&uid=swg24000647
- WITHDRAWN: SupportPac MA01: WebSphere MQ - Q program (q.exe)

- Even though MQ 8.0 and 9.0 are not mentioned in the list of supported versions, the SupportPac works well with MQ 8.0 and 9.0.

- It is NOT maintained by the MQ Support Team.

- Notice: This SupportPac has now been moved to GitHub and can be found here: https://github.com/ibm-messaging/mq-q-qload

Notes: SupportPac MA01

notes

- I have downloaded and extracted the zip file into my Windows box at:
- **C:\MQ-SupportPac\MA01 Q program**

- It is necessary to move to the directory for the platform:
- C:\MQ-SupportPac\MA01 Q program**V6.0\Windows**>

- The queue manager is remote.
- Need to setup MQSERVER:

set MQSERVER=ANGEL.SVRCONN/TCP/mosquito.raleigh.ibm.com(1424)

Notes: SupportPac MA01

notes

Run **q.exe** for the queue manager QMPS (-mQMPS),
using the remote network libraries (-lmqic)
(flag is lowercase letter l),
moving messages from queue (-IQSUBP1)
(flag is UPPERCASE letter I),
into queue (-oQSUBP2)(flag is lowercase letter o),
which have the specified CorrelId (-gxc) (flags are lowercase letters g destructive get, x hexadecimal and c correlationId)

```
q.exe -mQMPS  -lmqic -IQSUBP1 -oQSUBP2
-gxc414D5120514D50532020202020202089319C59E7080420
MQSeries Q Program by Paul Clarke [ V6.0.0 Build:May  1 2012 ]
Connecting ...connected to 'QMPS'.
No more messages.
```

Message was moved

- The q.exe program moved the message that was originally destined to QSUBP2, but that landed in QSUBP1, to the proper destination QSUBP2.
- Now QSUBP1 has 1 message (instead of 2)
- And QSUBP2 has 1 message (instead of 0)

Q1	Local	0	0	0
QSUBP1	Local	0	0	1
QSUBP2	Local	0	0	1
SYSTEM.ADMIN.ACCOUNTING.QUEUE	Local	0	0	0

Duplicating messages

- Some customers ask if the MQ queue manager has a facility to generate a duplicate message (for backup, archival or auditing reasons) for each message that arrives to a queue.

- The answer is NO.

- But there is a workaround that involves a
- Topic Object, a Topic Alias and
- Durable Subscriptions with Provided queues

Notes: Duplicating messages

notes

The following technote provides an approach that involves only administrative objects at the level of the queue manager, and this is transparent to the MQ Client applications. That is, the MQ client application is NOT affected at all (no changes!) and it is NOT aware that a duplicate is being generated by the queue manager.

NOTE: Strictly speaking, the messages received by the destination queues are NOT truly identical. Even though they will have the same payload and same characteristics, each message will have its own unique message-id.

.

http://www-01.ibm.com/support/docview.wss?uid=swg21574670

MQ: You want to put a message into a queue and you want to generate duplicate messages into other queues

Scenarios:

1) You are using point-to-point (queues) with your putting and getting applications. You want to use the MQ Pub/Sub under the covers and you do not want your putting and getting applications to be aware of Pub/Sub. That is, the application that puts messages into a queue continues to put messages into a queue, and the getting applications keep getting messages from a queue.

2) You are using point-to-point and you want to create "duplicate" messages that will be stored in queues Q1 and Q2, when the message is put into the queue "DESTINATION"

Notes: Duplicating messages

notes

** Baseline Scenario:
You have an MQ client application that puts a message into a queue called
DESTINATION
The queue DESTINATION will get 1 message for each 1 MQPUT.
Another MQ client application reads messages from the queue DESTINATION.

So far, so good.

** Modified Scenario:

However, now you get a requirement that a duplicate message needs to be generated.
When an MQ client application puts a message into DESTINATION you want to receive a message into:
Q1 (to be read by an application for normal processing) and
Q2 (to be handled by an application that deals with duplicate messages, for archival)

Notice that the MQ client application that used to read from the queue DESTINATION needs to read now from the queue Q1.

Notes: Duplicating messages

notes

++ Solution

Use MQ Explorer or amqsput sample to put a message to the DESTINATION queue, which via a topic alias, publishes the message into the topic T1 and the subscribers SUB1 and SUB2 receive the messages, which are stored in queues Q1 and Q2 respectively.
You could also publish directly into topic T1.
A copy of the message will be placed on Q1 and Q2.

Use runmqsc to define the following objects:

```
define qlocal(Q1)
define qlocal(Q2)
delete qlocal(DESTINATION)
define qalias(DESTINATION) target(T1) targtype(topic)
define topic(T1) topicstr('TOPIC1')
define sub(SUB1) topicstr('TOPIC1') dest(Q1)
define sub(SUB2) topicstr('TOPIC1') dest(Q2)
```

Notes: Duplicating messages

notes

++ Detailed scenario

The MQ client application keeps using MQPUT into the queue
DESTINATION
but it is no longer a local queue. It was deleted: **delete qlocal(DESTINATION)**
...and replaced by: **define qalias(DESTINATION) target(T1) targtype(topic)**

This means that when DESTINATION is used, instead of using a queue, the queue manager is going to use a topic called T1, which was defined as:
define topic(T1) topicstr('TOPIC1')

Then when a message arrives to DESTINATION, the queue manager will publish into topic T1 and a copy of the message will be sent to the subscribers:
define sub(SUB1) topicstr('TOPIC1') dest(Q1)
define sub(SUB2) topicstr('TOPIC1') dest(Q2)

These are durable subscribers that use "Provide" destination queues, already defined:
define qlocal(Q1)
define qlocal(Q2)

Running sample for durable subs: amqssbx

- The MQ sample "**amqssub**" uses non-durable subscriptions.

- If you want to experiment with durable subs, then you can use the MQ sample:
  - **amqssbx**

- You can still use "**amqspub**" to publish messages.

Notes: Using amqssbx to show resume

notes

- Open 2 command prompt windows:
  - Window 1: for amqspub
  - Window 2: for amqssbx

- ** Overall scenario:

- The publisher will publish 3 messages:
  - Minute-1: message-1
  - Minute-2: message-2
  - Minute-3: message-3

- The durable subscription application will receive the 3 messages.

- However, the application will take a 1 minute pause at Minute-2. That is, it will NOT be active when the publisher publishes message-2
  - Minute-1: application running and receives message-1
  - Minute-2: application is PAUSED (message-2 is stored in the destination queue)
  - Minute-3: application is RESUMES and receives both message-2 and message-3

Notes: Using amqssbx to show resume

notes

- Minute-0:
- Start client publisher application
- Start client subscriber application.

- Minute-1:
- Publish message-1, which will be received by running sub application.
- Do not publish for 35 seconds. The default for the amqssbx sample is to wait 30 seconds and then exit if messages are not received.
- Allow the sub application to terminate and exit (but keep the sub alive – taking a pause)

- Minute-2:
- Publish message-2. The sub application is NOT running, but message is going to be stored for the subscriber, in the destination queue.

- Minute-3:
- Restart the sub application (“resume”).
- The sub application will receive message-2.
- Publish message-3, which will be received by running sub application.

Notes: Using amqssbx to show resume

notes

- Window 2: Run durable subscriber SUB10 which uses provided queue Q3 for the topic string “sales/canada” and keep (do not delete) the sub when exiting (flag -k)
- **amqssbx -m QMPS -d SUB10 -q Q3 -t sales/canada –k**
- **Sample AMQSSBXA start**
- **Calling MQGET : wait for 30 seconds**

- Window 1: Publish message-1:
- **amqspub sales/canada QMPS**
- **Sample AMQSPUBA start**
- **target topic is sales/canada**
- **test-sales-canada-1**

- Window 2: Message-1 is received:
- **Message Properties matching 'mqps.%':**
- **MQTopicString : 'sales/canada'**
- **Message Data:**
- **'test-sales-canada-1'**
- **Calling MQGET : wait for 30 seconds**

Notes: Using amqssbx to show resume

notes

- Window 1: Do not publish for 35 seconds minute

- Window 2: Allow the sub application to end:
- **no more messages**
- **Sample AMQSSBXA end**

- Window 1: Publish message-2 (but sub application is not running)
- **test-sales-canada-2**

- Window 2: Resume sub application. Notice that message-2 is retrieved from the durable queue and displayed, then waits for next message:
- **amqssbx -m QMPS -d SUB10 -q Q3 -t sales/canada -k**
- Sample AMQSSBXA start
- Calling MQGET : wait for 30 seconds
- **Message Properties matching 'mqps.%':**
- **MQTopicString : 'sales/canada'**
- **Message Data:**
- **'test-sales-canada-2'**
- Calling MQGET : wait for 30 seconds

Notes: Using amqssbx to show resume

notes

- Window 1: Publish message-3
- **test-sales-canada-3**

- Window 2: Sub application receives message-3
- **Message Properties matching 'mqps.%':**
- **MQTopicString : 'sales/canada'**
- **Message Data:**
- **'test-sales-canada-3'**
- **Calling MQGET : wait for 30 seconds**

- **no more messages**
- **Sample AMQSSBXA end**

- Window 2:
- Just completeness, let's create a durable subscription but using a managed queue
- **amqssbx -m QMPS -d SUB11 -t sales/mexico**

Subscriptions view

- This updated list of subscriptions is shown below.

Subscription name	Topic name	Topic string	Destination class	Durable	Destination name	Type
		sales	Managed	No	SYSTEM.MANAGED.NDU...	API
QMPS SYSTEM.BROK...	SYSTEM.B...	SYSTEM.BR...	Provided	Yes	SYSTEM.BROKER.INTER.B...	API
SUB1		TOPIC1	Provided	Yes	Q1	Admin
SUB10		sales/canada	Provided	Yes	Q3	API
SUB11		sales/mexico	Managed	Yes	SYSTEM.MANAGED.DURA...	API
SUB2		TOPIC1	Provided	Yes	Q2	Admin
SUBM1	SALES	sales	Managed	Yes	SYSTEM.MANAGED.DURA...	Admin
SUBM2	SALES	sales	Managed	Yes	SYSTEM.MANAGED.DURA...	Admin
SUBM3		sales/mexico	Managed	Yes	SYSTEM.MANAGED.DURA...	Admin
SUBM4		sales/usa	Managed	Yes	SYSTEM.MANAGED.DURA...	Admin
SUBP1	SALES	sales	Provided	Yes	QSUBP1	Admin
SUBP2	SALES	sales	Provided	Yes	QSUBP2	Admin

Notes: Variety of subscriptions

notes

This presentation used a variety of subscriptions:
- Durable and non-durable
- API and Admin
- Managed queues and Provided queues

The following is from the Test Subscription dialog in MQ Explorer:

SubName	TopicName	TopicString	DestClass	Durable	DestQueue	Type
null	null	sales	Managed	No	S.M.NDurable.x	API

The following are from the amqssbx sample:

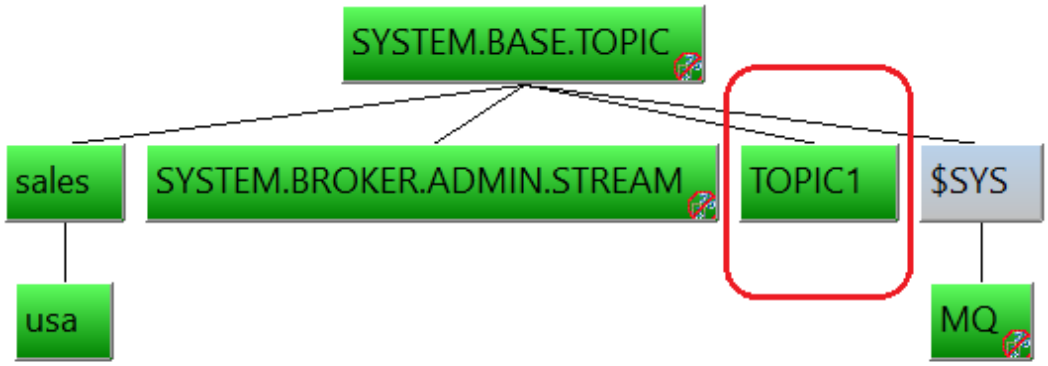
SubName	TopicName	TopicString	DestClass	Durable	DestQueue	Type
SUB10	null	sales/canada	Provided	Yes	Q3	API
SUB11	null	sales/mexico	Managed	Yes	S.M.Durable.x	API

The following are durable subs:

SubName	TopicName	TopicString	DestClass	Durable	DestQueue	Type
SUBM4	SALES	sales/usa	Managed	Yes	S.M.Durable.x	Admin
SUBP1	SALES	sales	Provided	Yes	QSUBP1	Admin

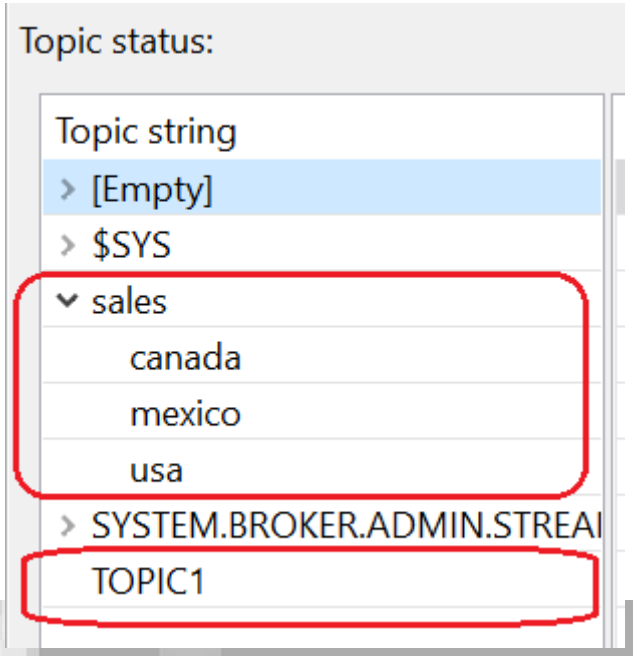
Topic Objects tree was changed

- The tree of the Topic Objects was changed.
- Notice the new addition: TOPIC1



Topic Status tree was changed

- The Topic Status tree was changed.
- Notice the new addition: TOPIC1
- And the non-durable topics for sales/canada and sales/mexico



The End

This is the end of the presentation.

THANKS!!