MQ Pub/Sub: Selectors, Retained Pubs, System Topics, Wildcards

http://www-01.ibm.com/support/docview.wss?uid=swg27050243

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

ON DEMAND BUSINESS

Related presentations

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 file

This techdoc has 1 zip file with files that are discussed in this presentation:

QMPS-trace-pub-sub-selectors.zip

The included files are:

- SampleJMSMsgProperty.class
- SampleJMSMsgProperty.java
- SimpleRetainedPub.class
- SimpleRetainedPub.java
- SYSTEM.RETAINED.PUB.QUEUE.contents.txt

Agenda

- Selectors
- Retained messages
- System Topics
- Wildcards

Subscribers with Selectors

Sometimes a Topic is very busy with many published messages.
Some customers would like to filter down the messages received by a subscriber, by using a **SELECTOR (which uses a message property).**

The publisher may add a particular value for a message property.
The subscriber will get a message only if the specified Selector matches the value of the message property in the published message.

Subscribers with Selectors

For example, lets define 3 durable subscribers for the same topic string “sales”, with provided queues.

1) For all published messages, regardless of the value for the message property: color
Sub: SALES Provided Q: QSALES

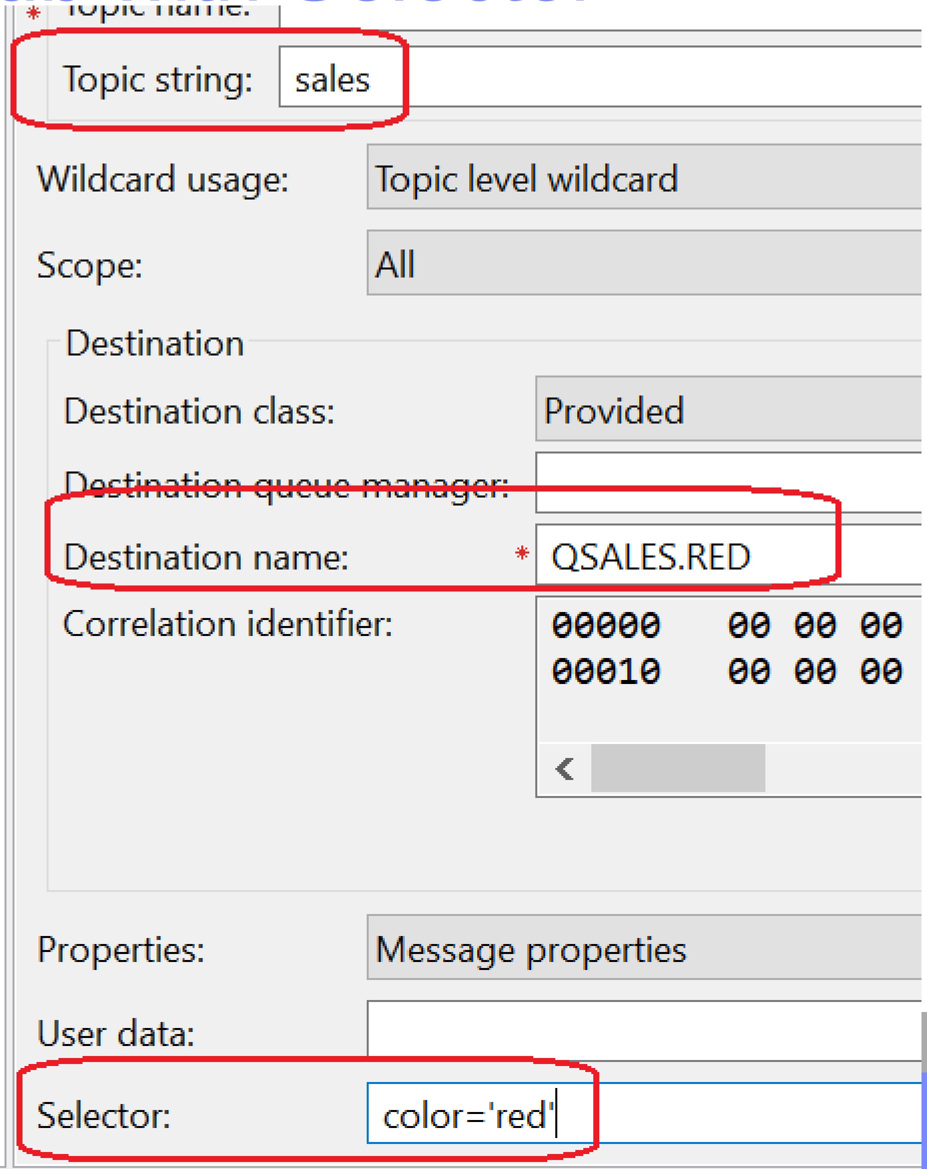
2) Only when the message property: **color=‘red’**
Sub: **SALES.RED** Provided Q: QSALES.RED

3) Only when the message property: **color=‘blue’**
Sub: SALES.BLUE Provided Q: QSALES.BLUE

MQ Explorer, define Sub with Selector

When using MQ Explorer it is straight forward to define a durable subscriber with a selector.
For SALES.RED:
There is NO need to include extra quotes for the selector.

color='red'



Topic name:
Topic string: sales
Wildcard usage: Topic level wildcard
Scope: All
Destination
Destination class: Provided
Destination queue manager:
Destination name: * QSALES.RED
Correlation identifier: 00000 00 00 00
00010 00 00 00
Properties: Message properties
User data:
Selector: color='red'

runmqsc, subscribers with Selectors

+++ CAVEAT when using runmqsc and the SELECTOR attribute!!!

During the DEFINE statement, it is CRITICAL to ensure that the whole string for the selector, which specifies the name-value pair, has a leading single quote and a trailing single quote:

```
SELECTOR('name=value')
         *          *
```

runmqsc, subscribers with Selectors

But when the value is a **string**, then this becomes tricky, because the string needs to be placed between:
- a leading set of 2 single quotes and
- a trailing set of 2 single quotes.

Example:
```
''red''
**   **
```

Common pitfall:
Do NOT enter a single double quote to replace the 2 single quotes!

runmqsc, subscribers with Selectors

To show that it is very easy to get confused, the following examples use a **proportional font**:

Correct (using 2 single quotes to surround the value red):

SELECTOR('color=''red''')

Incorrect (using double quotes to surround the value red):

SELECTOR('color="red"')

runmqsc, subscribers with Selectors

When using a **monospace font**, it will be easier to see that the correct version does NOT use a double quote, but a pair of single quotes:

Correct (using 2 single quotes)

```
SELECTOR('color=''red''')
             **    **
```

Incorrect (using double quotes)

```
SELECTOR('color="red"')
             *   *
```

runmqsc, subscribers with Selectors

Example:
Notice that the + at the end of each line indicates that the statement continues in the next line:

```
define sub(SALES.RED)+
 topicstr('sales') dest(QSALES.RED)+
 SELECTOR('color=''red''')
```

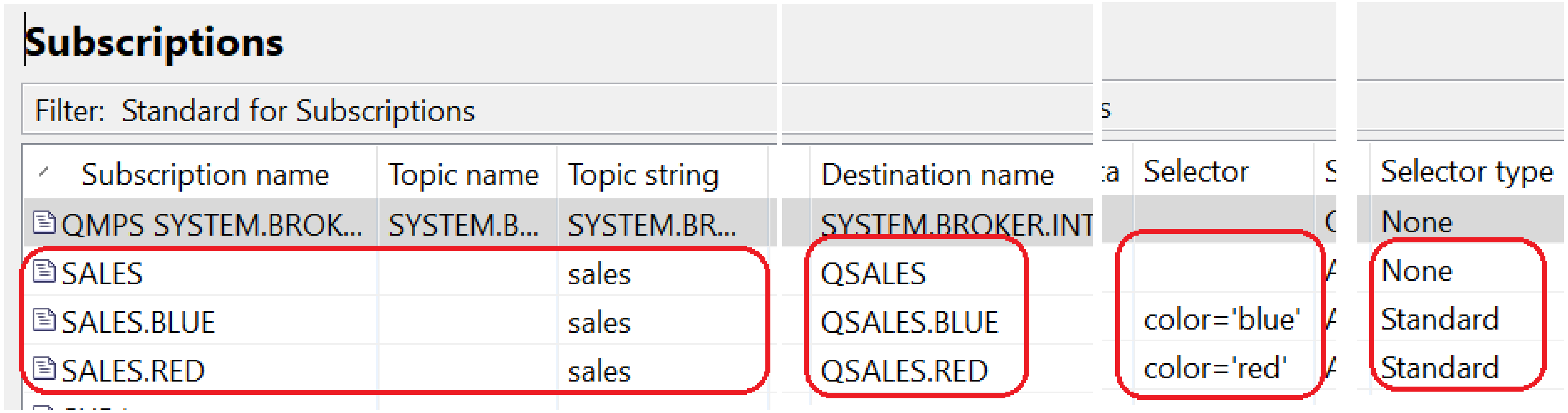
runmqsc, subscribers with Selectors

Verify that the SELECTOR is properly shown.
When displaying the attributes, the leading and trailing single quotes are NOT shown.
These extra single quotes are ONLY needed during a DEFINE

```
display sub(SALES.RED) selector
AMQ8096: WebSphere MQ subscription inquired.
   SUB(SALES.RED)              SELECTOR(color='red')
```

MQ Explorer, subscribers with Selectors

MQ Explorer view of these subscriptions.
Showing a composite view.
Notice the Selector Type column.



Subscriptions

Filter: Standard for Subscriptions

Subscription name	Topic name	Topic string	Destination name	Selector	Selector type
QMPS SYSTEM.BROK...	SYSTEM.B...	SYSTEM.BR...	SYSTEM.BROKER.INT		None
SALES		sales	QSALES		None
SALES.BLUE		sales	QSALES.BLUE	color='blue'	Standard
SALES.RED		sales	QSALES.RED	color='red'	Standard

Sample source code to include properties

The sample "SampleJMSMsgProperty.java" can be used to publish a message with or without properties

* To produce a message with out the message property "color"

* **java SampleJMSMsgProperty**

* To produce a message with the message property "color='red'"

* **java SampleJMSMsgProperty red**

Notice the specification with 2 forward slashes:

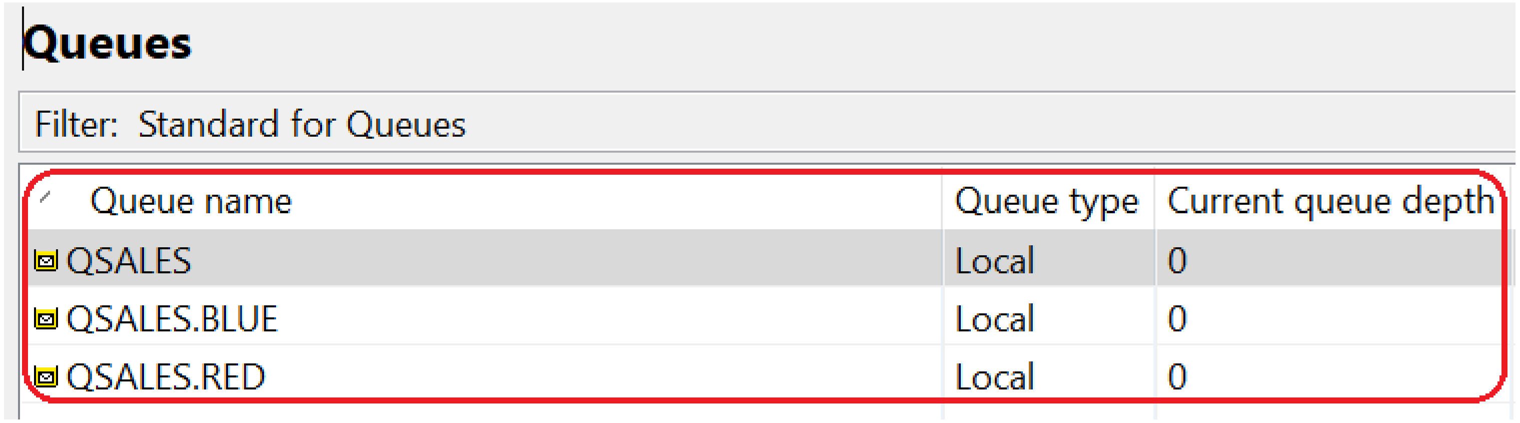
```
destination = session.createTopic("topic://sales");
```

Notice the setStringProperty method:

```
if (color == null) {
   System.out.println("* Creating message with no color");
 } else {
   System.out.println("* Creating message with color " + color);
   message.setStringProperty("color", color);
 }
```

Viewing the Provided queues

The provided queues have a CURDEPTH (the Current Queue Depth) of 0.
That is, no messages have been published.



Queues

Filter: Standard for Queues

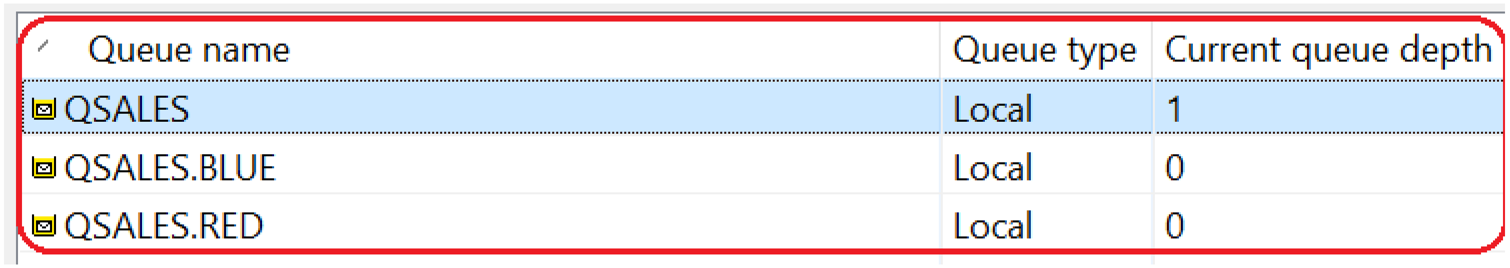
Queue name	Queue type	Current queue depth
QSALES	Local	0
QSALES.BLUE	Local	0
QSALES.RED	Local	0

Viewing the Provided queues

Publishing 1 message **without** a property.

```
java SampleJMSMsgProperty
```

Notice that ONLY the queue QSALES received a message.
The other queues did NOT because their Selectors did not match the message property.



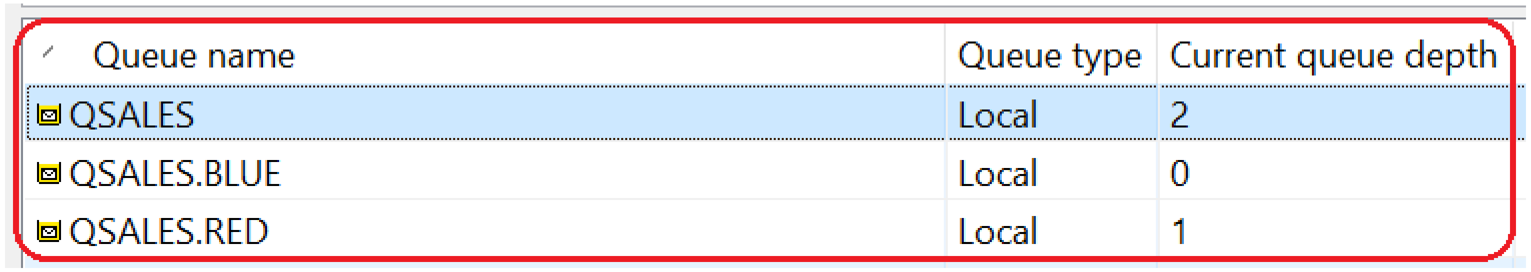
Queue name	Queue type	Current queue depth
QSALES	Local	1
QSALES.BLUE	Local	0
QSALES.RED	Local	0

Viewing the Provided queues

Publishing 1 message **with** a property.

java SampleJMSMsgProperty **red**

Notice that QSALES received the 2nd message, because this queue will get ALL.
QSALES.RED received the message.
While QSALES.BLUE did NOT receive.



Queue name	Queue type	Current queue depth
QSALES	Local	2
QSALES.BLUE	Local	0
QSALES.RED	Local	1

Retained messages

By default, after a publication is sent to all interested subscribers it is discarded.

A publisher can specify that **a copy of a publication is retained** so that it can be sent to future subscribers who register an interest in the topic.

Deleting publications after they have been sent to all interested subscribers **is suitable for event information**,
but is not always suitable for state information.

Retained messages

By retaining a message, new subscribers
do not have to wait for information to be published
again before they receive initial state information.
For example, a new subscriber for a stock price
would receive the current price straight away,
without waiting for the stock price to change (and be
republished).

Only one publication for each topic is retained,
so the existing retained publication of a topic is
deleted when a new retained publication arrives.

Retained messages, sample

The following sample source code is available:

..\jms\sample\simple\SimpleRetainedPub.java

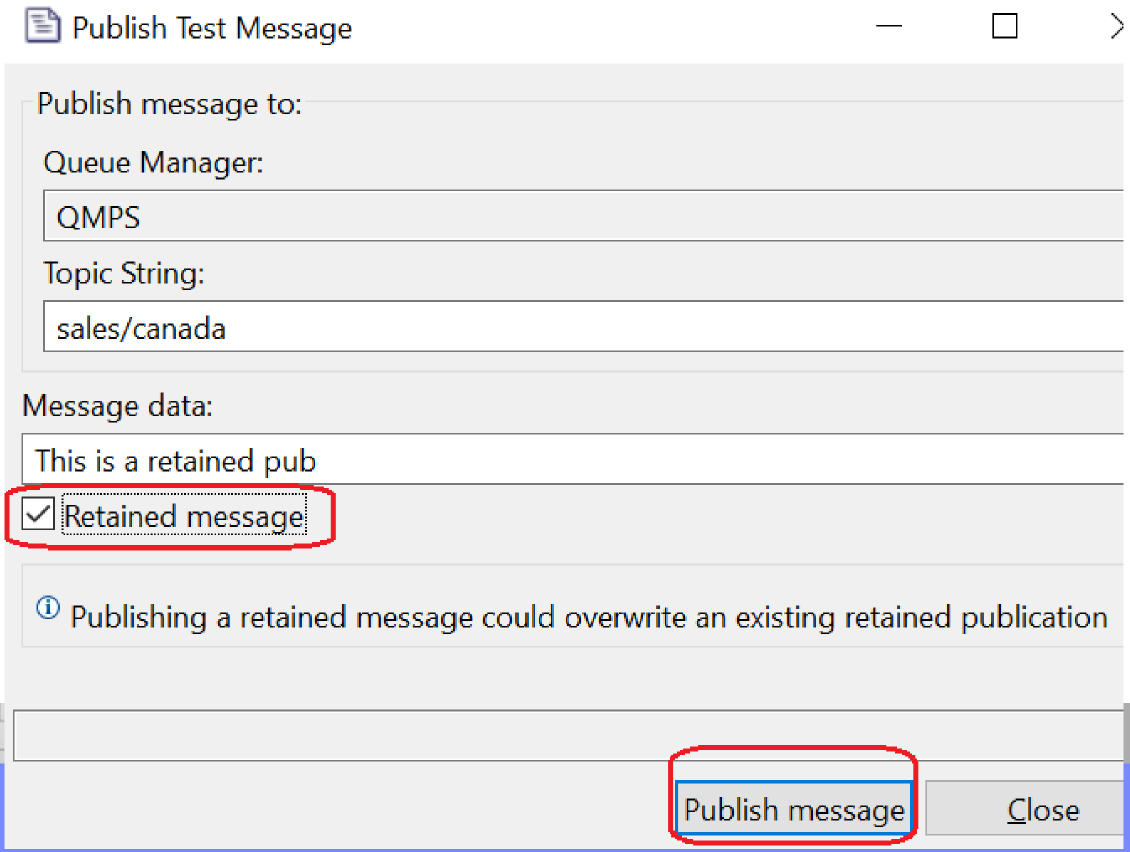
This is the code that enables the function:

```
// Mark the message as a retained publication
message.setIntProperty(WMQConstants.JMS_IBM_RETAIN,
                       WMQConstants.RETAIN_PUBLICATION);
```

A customized copy for topic “sales/canada” is included in the zip file for this presentation.
To execute it: java SimpleRetainedPub

Retained messages with MQ Explorer

From "Publish Test Message" enable the checkbox (*) Retained message



Publish Test Message

Publish message to:

Queue Manager: QMPS

Topic String: sales/canada

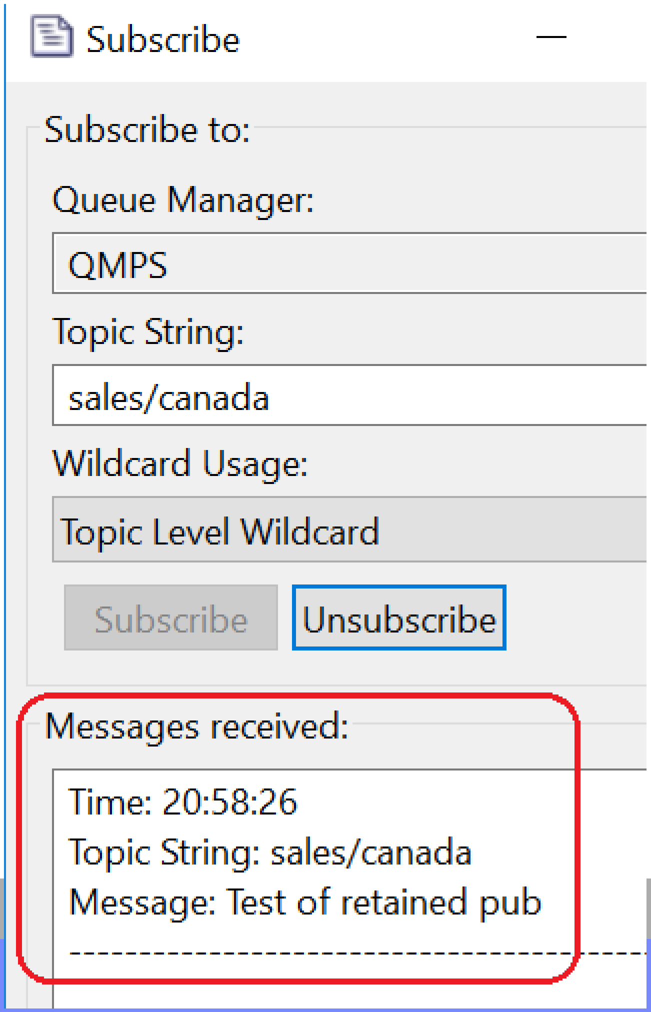
Message data: This is a retained pub

☑ Retained message

Publishing a retained message could overwrite an existing retained publication

Retained messages with MQ Explorer

When opening a new "Test Subscription" for the topic string, the retained publication is delivered to the subscriber.



Subscribe

Subscribe to:

Queue Manager: QMPS

Topic String: sales/canada

Wildcard Usage: Topic Level Wildcard

Messages received:

```
Time: 20:58:26
Topic String: sales/canada
Message: Test of retained pub
--------------------------------------------
```

How to see which topics have retained msg

Using **DISPLAY TPSTATUS**.

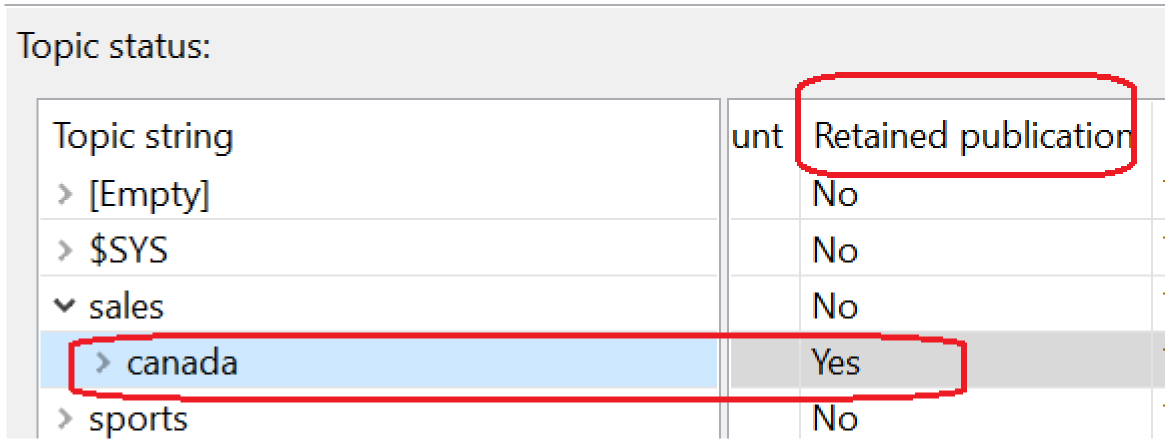
The **RETAINED** field shows whether there is a retained message.

```
DISPLAY TPSTATUS('#') TYPE(TOPIC) WHERE(RETAINED EQ YES)
```

```
AMQ8754I: Display topic status details.
   TOPICSTR(sales/canada)     RETAINED(YES)
```

How to see which topics have retained msg

In MQ Explorer, open the Topic Status,
then select the desired topic string.
On the right panel, scroll to the right until to see the column titled “Retained publication”.



Topic status:

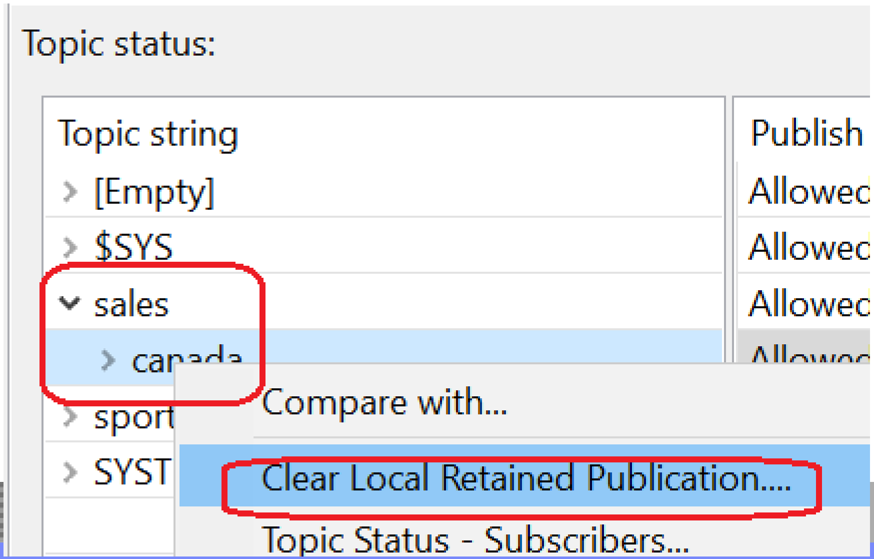
Topic string	unt	Retained publication
[Empty]		No
$SYS		No
sales		No
canada		Yes
sports		No

How to stop using a retained publication

To stop using a retained publication for a topic:

1) runmqsc: **CLEAR TOPICSTR**

2) MQ Explorer: show Topic Status, then select desired topic, then "Clear Local Retained Publication"



Topic status:

Topic string	Publish
[Empty]	Allowed
$SYS	Allowed
sales	Allowed
canada	Allowed
sport	
SYST	

Compare with...
Clear Local Retained Publication....
Topic Status - Subscribers...

Retained messages System Q / Sample

The retained messages are stored in the following System queue:

SYSTEM.RETAINED.PUB.QUEUE

Let's show how many messages are in the queue:

```
DISPLAY QL(SYSTEM.RETAINED.PUB.QUEUE) CURDEPTH
AMQ8409I: Display Queue details.
   QUEUE(SYSTEM.RETAINED.PUB.QUEUE)       TYPE(QLOCAL)
   CURDEPTH(25)
```

Wow! 25!!??

We specified to retain only for 'sales/canada', what is going on?

Notes: Looking at the retained messages

notes

We can use the following MQ sample to browse the contents of the queue that has the retained messages (it is a long command that needs to be executed it in 1 line, but showing it below in 2 lines for readability)

```
amqsbcg SYSTEM.RETAINED.PUB.QUEUE QMPS >
          SYSTEM.RETAINED.PUB.QUEUE.contents.txt
```

Looking at the output file, let's search for 'sales' and we find a normal looking JMS message that has an RFH2 header and inside, the topic string:

```
00000120:  7372 3E20 2400 0000 3C6D 7170 733E 3C54 'sr> $...<mqps><T'
00000130:  6F70 3E73 616C 6573 2F63 616E 6164 613C 'op>sales/canada<'
00000140:  2F54 6F70 3E3C 2F6D 7170 733E 5369 6D70 '/Top></mqps>Simp'
00000150:  6C65 5265 7461 696E 6564 5075 623A 2059 'leRetainedPub: Y'
00000160:  6F75 7220 6C75 636B 7920 6E75 6D62 6572 'our lucky number'
00000170:  2074 6F64 6179 2069 7320 3136 37        ' today is 167   '
```

OK, this is 1 out of the 25 messages, what about the others?

Notes: Looking at the retained messages

n o t e s

A sample of a couple of the other messages reveal a pattern:

One sample. Notice the topic string:
$SYS/MQ/INFO/QMGR/QMPS/Monitor/CPU/SystemSummary

```
…
00000160:  B804 0000 3000 0000 2453 5953 2F4D 512F '....0...$SYS/MQ/'
00000170:  494E 464F 2F51 4D47 522F 514D 5053 2F4D 'INFO/QMGR/QMPS/M'
00000180:  6F6E 6974 6F72 2F43 5055 2F53 7973 7465 'onitor/CPU/Syste'
00000190:  6D53 756D 6D61 7279                     'mSummary        '
```

Another sample. Notice another topic string:
$SYS/MQ/INFO/QMGR/QMPS/Monitor/CPU/QmgrSummary

```
..
00000210:  2E00 0000 2453 5953 2F4D 512F 494E 464F '....$SYS/MQ/INFO'
00000220:  2F51 4D47 522F 514D 5053 2F4D 6F6E 6974 '/QMGR/QMPS/Monit'
00000230:  6F72 2F43 5055 2F51 4D67 7253 756D 6D61 'or/CPU/QMgrSumma'
00000240:  7279 0000                               'ry..             `
```

What are these $SYS topics? It will be explained shortly!

Notes: System Topics (new in MQ 9.0)

n o t e s

MQ 9.0 introduced a new feature:

https://www.ibm.com/support/knowledgecenter/SSFKSJ_9.0.0/com.ibm.mq.mon.doc/mo00040_.htm
IBM MQ 9.0 > Monitoring and performance >
Monitoring your IBM MQ network >
System topics for monitoring and activity trace

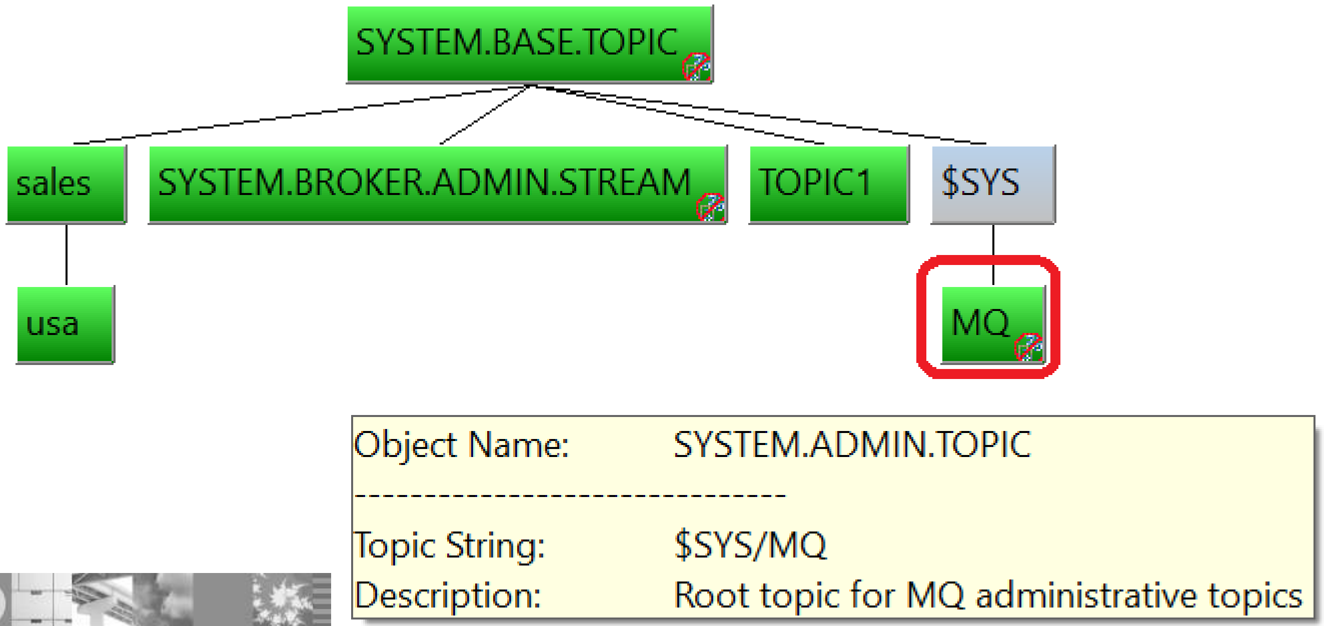
System topics in queue manager topic trees are used for resource monitoring and for application activity trace.

Each queue manager's topic tree contains the $SYS/MQ branch.
The queue manager publishes to topic strings in this branch every 10 seconds.

An authorized user can subscribe to these topic strings to receive information on the queue manager and the activity on it.

Tree structure of System Topics

The System Topics has a tree structure that is based on the topic string $SYS/MQ which is defined under the topic object SYSTEM.ADMIN.TOPIC



Object Name: SYSTEM.ADMIN.TOPIC

Topic String: $SYS/MQ
Description: Root topic for MQ administrative topics

Notes: System Topics (new in MQ 9.0)

n o t e s

An example of a System Topic that you can subscribe to is:

$SYS/MQ/INFO/QMGR/QMPS/Monitor/**CPU/QMgrSummary**

The following is the tree of the System Topics:

CPU : Platform central processing units
SystemSummary : CPU performance - platform wide
QMgrSummary : CPU performance - running queue manager

DISK : Platform persistent data stores
SystemSummary : Disk usage - platform wide
QMgrSummary : Disk usage - running queue managers
Log : Disk usage - queue manager recovery log

STATMQI : API usage statistics

CONNDISC : MQCONN and MQDISC	OPENCLOSE : MQOPEN and MQCLOSE
INQSET : MQINQ and MQSET	PUT : MQPUT
GET : MQGET	SYNCPOINT : Commit and rollback
SUBSCRIBE : Subscribe	PUBLISH : Publish

STATQ : API per-queue usage statistics

OPENCLOSE : MQOPEN and MQCLOSE	INQSET : MQINQ and MQSET
PUT : MQPUT and MQPUT1	GET : MQGET

System Topics

The following pages are based on the following tutorial:

http://www.ibm.com/support/docview.wss?uid=swg27049331
Using MQ 9.0 system topics for resource monitoring and MQ 9.0.1 Console for showing charts

Notes:

n o t e s

To test the publishing of messages for the system topics:

- Local queue to received the messages:

QSYS1 => to be used with subscriber SYS1 (CPUQMgrSummary)

- Subscription:

It receives the messages in the provided queue QSYS1

define sub(SYS1) dest(QSYS1) destclas(provided) +
topicstr('$SYS/MQ/INFO/QMGR/QMPS/Monitor/CPU/QMgrSummary')

Note: The subject of wildcards will be explained shortly.

You CANNOT use wildcards on the $SYS topic tree!

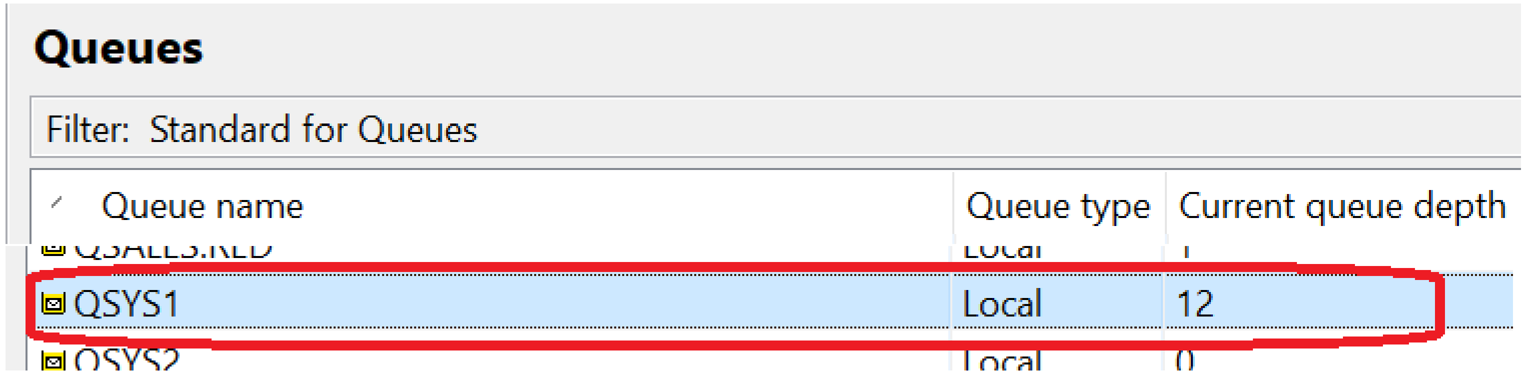
For example:

define sub(S3) dest(QS3) destclas(provided) topicstr('**$SYS/MQ/INFO/QMGR/QMPS/Monitor/#**')
AMQ8101E: IBM MQ error (A26) has occurred.

Notes:

notes

The queue manager publishes updates to the System Topics every 10 seconds.
ATTENTION! You need to plan to consume the published messages, otherwise they could quickly fill up the provided queue!
After just more than a minute notice that there are 12 messages!

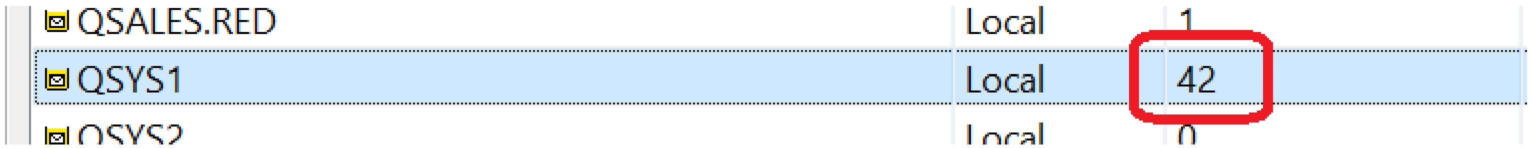


Queues

Filter: Standard for Queues

Queue name	Queue type	Current queue depth
QSALES.RED	Local	1
QSYS1	Local	12
QSYS2	Local	0

And after few more minutes notice the increase to 42:



QSALES.RED	Local	1
QSYS1	Local	42
QSYS2	Local	0

Notes:

notes

What do the messages look like? Format **MQPCF**, put by the queue manager

```
  StrucId  : 'MD  '  Version : 2
  Report   : 0  MsgType : 8
  Expiry   : 35934  Feedback : 0
  Encoding : 546  CodedCharSetId : 437
  Format : 'MQPCF   '
  Priority : 0  Persistence : 0
  MsgId :    X'414D5120514D505320202020202020205D6EB4592380B37C'
...
  ** Origin Context
  PutApplType     : '26'
  PutApplName     : 'QMPS                        '
  PutDate  : '20170911'    PutTime   : '14373710'
...
00000000:  1500 0000 2400 0000 0300 0000 0000 0000 '....$...........'
00000010:  0100 0000 0100 0000 0000 0000 0000 0000 '................'
00000020:  0700 0000 0400 0000 4400 0000 DF07 0000 '........D.......'
00000030:  5203 0000 3000 0000 514D 5053 2020 2020 'R...0...QMPS    '
00000040:  2020 2020 2020 2020 2020 2020 2020 2020 '                '
00000050:  2020 2020 2020 2020 2020 2020 2020 2020 '                '
00000060:  2020 2020 2020 2020 0300 0000 1000 0000 '        ........'
00000070:  4703 0000 0000 0000 0300 0000 1000 0000 'G...............'
00000080:  4803 0000 0100 0000 1700 0000 1800 0000 'H...............'
00000090:  4D03 0000 0100 0000 2AB0 9800 0000 0000 'M.......*.ÿ.....'
000000A0:  0300 0000 1000 0000 0000 0000 0200 0000 '................'
000000B0:  0300 0000 1000 0000 0100 0000 0300 0000 '................'
000000C0:  0300 0000 1000 0000 0200 0000 9700 0000 '............ù...'
```

Notes: sample amqsrua

notes

Example of using the MQ sample that displays every 10 seconds the messages from system object that ends with: …/CPU/QmgrSummary

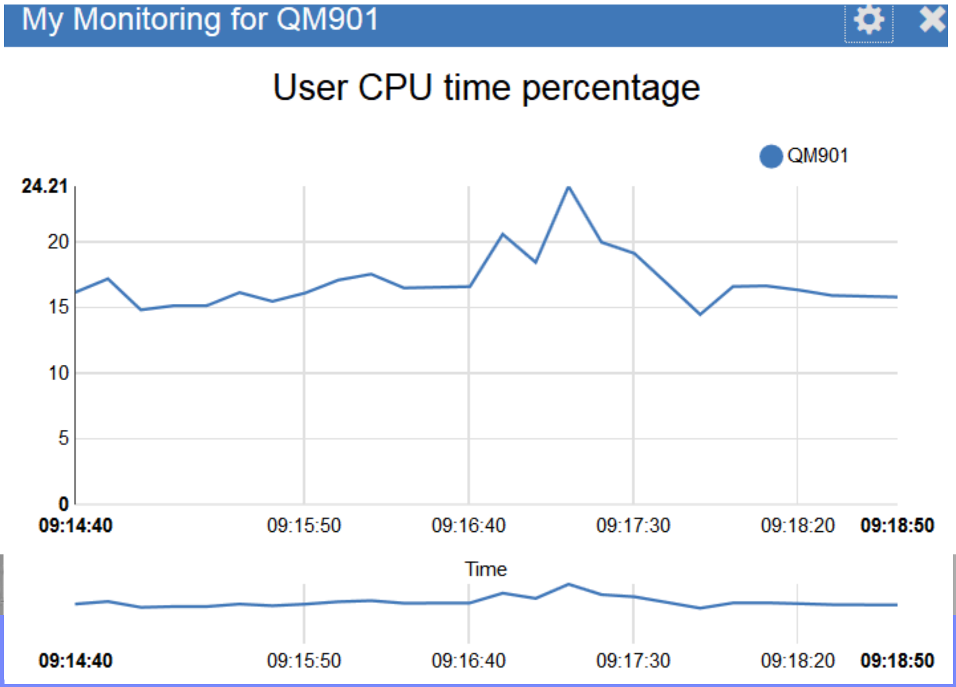
Notice that the sample shows the FIRST message very quickly!
Why? Because this first message is a RETAINED publication and it is available immediately to the subscriber!

```
$ amqsrua -n 10 -m QMPS -c CPU -t QMgrSummary
Publication received PutDate:20170911 PutTime:14413570 Interval:8.468 seconds
User CPU time - percentage estimate for queue manager 0.01%
System CPU time - percentage estimate for queue manager 0.02%
RAM total bytes - estimate for queue manager 151MB

Publication received PutDate:20170911 PutTime:14414570 Interval:10.004 seconds
User CPU time - percentage estimate for queue manager 0.01%
System CPU time - percentage estimate for queue manager 0.04%
RAM total bytes - estimate for queue manager 151MB
```

MQ Console (9.0.1)

MQ 9.0.1 introduced a new feature: the MQ Console.
It is a web browser interface to administer MQ queue managers.
It can be used to show charts from System Topics!



Wildcards not supported in $SYS/MQ

When I was planning the contents for this presentation, I was going to use the System Topics tree ($SYS/MQ) to introduce the subject of:

Wildcards

Well, when I tried to create a subscriber with Wildcards under $SYS/MQ I got an error message. I checked the online manual and it is working as designed!

You cannot use a wildcard under $SYS/MQ !!

Wildcards

Topic-based wildcards allow subscribers to subscribe to more than one topic at a time.

There are 2 types:

Multilevel Wildcard => #

It is used to match any number of levels within a topic.

Single Level Wildcard => +

It matches one, and only one, topic level.

For more information see:
https://www.ibm.com/support/knowledgecenter/SSFKSJ_9.0.0/com.ibm.mq.pro.doc/q005020_.htm
IBM MQ > Technical overview > Publish/subscribe messaging > Publish/subscribe components > Topics > Topic strings > Wildcard schemes

Tree hierarchy of sales/country/state

Instead of the $SYS/MQ tree, I am going to use the hierarchy used in previous presentations:

```
sales                    sales
  |                  +-----|---+
  |                  |         |
country            canada     usa
  |                      +-----|---+
  |                      |         |
state                  alaska    texas
```

Notes: Topic-based wildcard scheme

notes

Based on our test hierarchy, the following test cases for a topic string will be done:
sales
sales/canada
sales/usa
sales/usa/alaska
sales/usa/texas

The following table shows what matches and what does not match a given subscriber that uses a wildcard:

(subscriber ->)	Multiple Level sales/#	Single Level sales/+	Multiple Level sales/usa/#
(Topic string)			
sales	yes		
sales/canada	yes	yes	
sales/usa	yes	yes	yes
sales/usa/alaska	yes		yes
sales/usa/texas	yes		yes
(messages received)	5	2	3

Subscribers for the sales topic tree

We will use 3 subscribers to represent each of the examples from the previous page.
Each subscriber will use a provided queue which has a corresponding name.

MultiLevel Wildcard:
define sub(**SALES.MLW**) dest(Q.SALES.MLW) destclas(provided) +
topicstr(**'sales/#'**)

MultiLevel Wildcard:
define sub(**SALES.USA.MLW**) dest(Q.SALES.USA.MLW) destclas(provided) +
topicstr(**'sales/usa/#')**

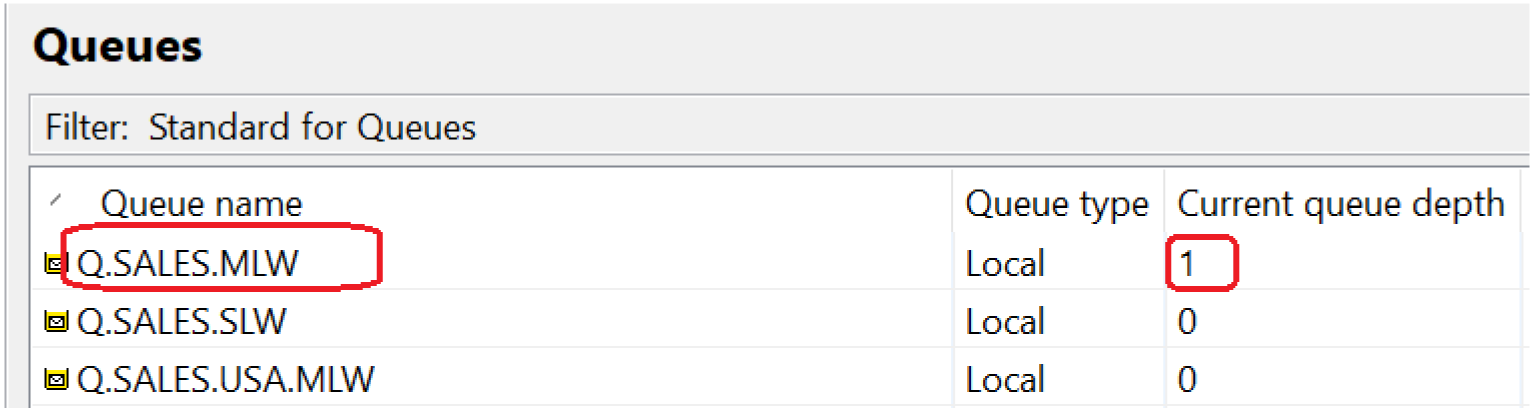
Single Level Wildcard:
define sub(**SALES.SLW**) dest(Q.SALES.SLW) destclas(provided) +
topicstr(**'sales/+'**)

Subscribers for the sales topic tree

Use MQ Explorer or amqspub to publish 1 message for each of the test topic strings.

Test 1: sales

	sales/#	sales/+	sales/usa/#
sales	yes		



Queues

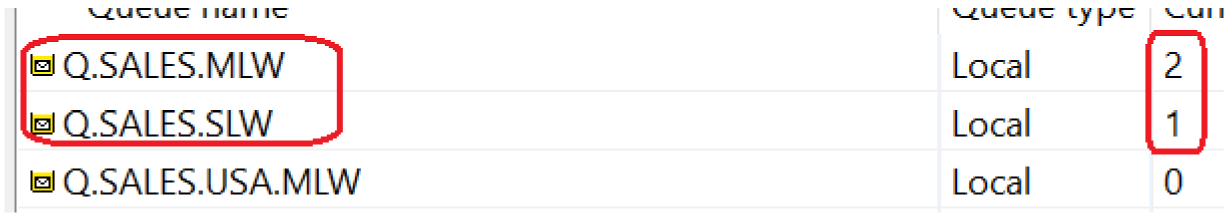
Filter: Standard for Queues

Queue name	Queue type	Current queue depth
Q.SALES.MLW	Local	1
Q.SALES.SLW	Local	0
Q.SALES.USA.MLW	Local	0

Subscribers for the sales topic tree

Test 2: sales/canada

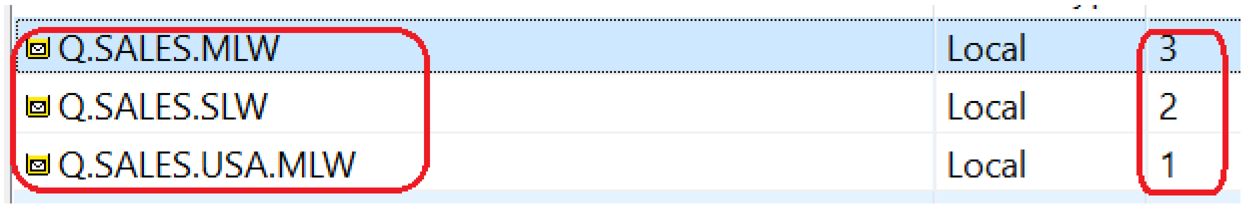
	sales/#	sales/+	sales/usa/#
sales/canada	yes	yes	



Q.SALES.MLW	Local	2
Q.SALES.SLW	Local	1
Q.SALES.USA.MLW	Local	0

Test 3: sales/usa

	sales/#	sales/+	sales/usa/#
sales/usa	yes	yes	yes

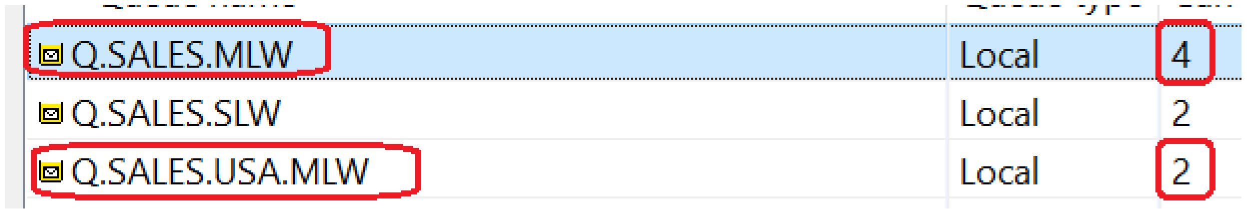


Q.SALES.MLW	Local	3
Q.SALES.SLW	Local	2
Q.SALES.USA.MLW	Local	1

Subscribers for the sales topic tree

Test 4: sales/usa/alaska

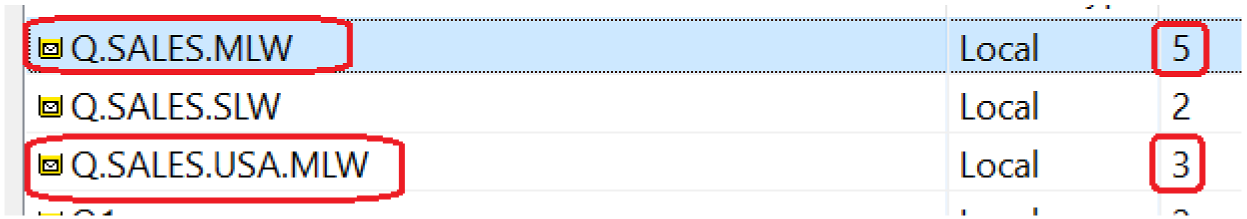
	sales/#	sales/+	sales/usa/#
sales/usa/alaska	yes		yes



Q.SALES.MLW	Local	4
Q.SALES.SLW	Local	2
Q.SALES.USA.MLW	Local	2

Test 5: sales/usa/texas

	sales/#	sales/+	sales/usa/#
sales/usa/texas	yes		yes



Q.SALES.MLW	Local	5
Q.SALES.SLW	Local	2
Q.SALES.USA.MLW	Local	3

Examples of using wildcard # in runmqsc

In runmqsc you can use the topic wildcards.
Some examples are:

1) Page 24 in this presentation:

```
DISPLAY TPSTATUS('#') TYPE(TOPIC) WHERE(RETAINED EQ YES)
```

2) The runmqras uses it when specifying "-section defs":

(cd to directly where the files from runmqras were unzipped)

```
$ cd defs
$ grep # *
runmqsc_TPSTATUS_PUB_QMPS.stdout:    1 : DIS TPSTATUS('#') TYPE(PUB)   ALL
runmqsc_TPSTATUS_SUB_QMPS.stdout:    1 : DIS TPSTATUS('#') TYPE(SUB)   ALL
runmqsc_TPSTATUS_TOPIC_QMPS.stdout: 1 : DIS TPSTATUS('#') TYPE(TOPIC) ALL
```

The End

This is the end of the presentation.

THANKS!!