



Wildfire Workshop
Washington Systems Center
Technical Hands-On
Workshops

Queue Level Statistics!!!!

Lyn Elkins

elkinsc@us.ibm.com

Dorothy Quincy

Dorothy.quincy@ibm.com



A bit of background

There has never been anything more fraught than MQ for z/OS SMF. From misleading numbers, to misleading reports, to We've seen it all. What we have also seen is that it is getting better. The development lab is paying attention, customers are realizing how important the data is, and the costs of collection have come down.



YOUR SMF
EXPERIENCE WILL
VARY

Agenda

- Queue Statistics - Introduction
- Queue Statistics – V9.3.3!!!
- Will this replace Task accounting data?
- Looking for volunteers!



Warning!!!!

Grab lots of coffee.....

Queue Statistics - Introduction

- History of Best Practices:
 - When MQ was an infant/toddler
 - Often the advice was to use as few queues as possible
 - Keep one application's queues isolated from others or keeping all transmission queues in one storage class.
 - Isolate the `SYSTEM.CLUSTER.TRANSMIT.QUEUE (SCTQ)`
 - This is still true, but should be 'whenever possible isolate any cluster transmission queues'
 - Get the SCTQ off BP 0 if it is still there.
 - "Know thy Cluster" was challenging for new admins, even for experienced personnel when one part of the cluster starts misbehaving.
 - The only way to get meaningful information about queue use was:
 - Monitoring tools that issue `DISPLAY` and `RESET QSTATS` commands.
 - Getting Accounting Class 3 – or Task Accounting data
 - Lots of resistance to this for all the wrong reasons.



10 years
ago (15?)

- I asked the development lab for queue statistics data, because:
 - Pushback on cost of class 3 accounting
 - Some of this was justified.
 - MQ Admins may not see workload growth in particular applications if the queues are not isolated.
 - Even applications folks can get surprised!
 - Application queues, even when isolated, may not show where workload changes have happened.

My request evolved over the years.....

- My first request for queue stats was simply a reproduction of the data from the two monitoring commands commonly used:
 - DISPLAY QSTATUS
 - RESET QSTATS
- This was not easily 'doable' because of RESET QSTATS
- And both the product, its use, and the critical nature of the data have evolved.
- My last request was several pages long but could be summarized by 'give us most of what shows up in the WQ records, but by statistics interval'

[This Photo](#) by Unknown Author is licensed under [CC BY-SA](#)



Queue Statistics!!!!

- As of MQ 9.3.0 there are some basic queue statistics provided
 - First version (9.3.0) laid the foundation
 - Second version from the CD stream (9.3.1) expanded it slightly
 - Third version from the CD stream 9.3.3 expanded it a LOT more!
 - And those new fields will be described in this session.

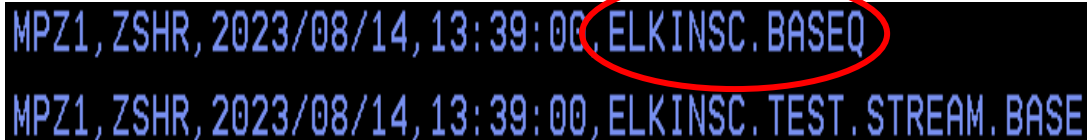
We (the WSC) do not have production data from any customer to date for this. Our current interpretation may change after we have seen this from some of the more active queue managers on z/OS.

Queue Statistics as of 9.3.3

- Quite possibly the best documented record in the collection of MQ SMF data. (Thank you Ed!) Please see:
 - <https://www.ibm.com/docs/en/ibm-mq/9.3?topic=statistics-queue-data-records-version-933-release>
- There will be customers that will want to install the CD version for this capability.

Standard Data:

- QQSTID – Record Identifier - x 'D80F'
- QQSTLL – Control Block Length
- QQSTEYEC – QQST
- QQSTQNAM – The queue name.
 - Please note that this is the base name, not the queue open name:



MPZ1,ZSHR,2023/08/14,13:39:00,ELKINSC.BASEQ
MPZ1,ZSHR,2023/08/14,13:39:00,ELKINSC.TEST.STREAM.BASE

The image shows a screenshot of two lines of data. The first line is "MPZ1,ZSHR,2023/08/14,13:39:00,ELKINSC.BASEQ" and the second line is "MPZ1,ZSHR,2023/08/14,13:39:00,ELKINSC.TEST.STREAM.BASE". The text "ELKINSC.BASEQ" in the first line is circled in red, highlighting the queue name.

- QQSTFLAG – Bit flags that provide information about the queue, to date:
 - Whether the queue is private or a shared queue
 - If this is a partial record – can happen when some info not available
 - Uncommitted messages on queue (either MQPUTs or MQGETs)

Fields about the queue that are of interest:

- QQSTPSID – Private Queue Page Set ID
 - If a shared queue this will be -1
- QQSTBPID – Buffer Pool ID
 - If a shared queue this will be -1
- QQSTQSGN – Queue Sharing Group Name
 - This is Blanks if a private queue
- QQSTDPTH – Queue current depth, **NOT THE HIGH WATER MARK!!**
 - Same as from DISPLAY QSTATUS
- Open Counts:
 - QQSTOPCT* – Number of applications with the queue open for output
 - QQSTIPCT* - Number of applications with the queue open for input
 - * - These are the current counts and only reflect the local connections for shared queues.

Fields about the queue that are of interest:

- QQSTMAGE – Age of oldest message on the queue
 - Can be 0 (Zero) if this is a partial record
 - Can be used to see if there may be old, unprocessed messages on a queue
 - Same Value as from DISPLAY QSTATUS
- **Queue Depths for interval:**
 - **QQSTDPHI** – Highest queue depth during this interval
 - **QQSTDPL0** - Lowest queue depth during this interval
 - Both of these fields are based on what has been done in this queue manager, not totals for the QSG is this is a shared queue.
 - This may be some of the most valuable information, as we have had to rely on Accounting Class 3 data to see highest and lowest depths on a queue

API MQPUT and PUT1 Data!!!

- For shared queues, these numbers just reflect the API requests for this queue manager
- QQSTPUTS – Number of MQPUT requests for the interval
- QQSTPUT1 – Number of MQPUT1 requests for the interval
- QQSTNPPT & QQSTNPP1 – Number of non-persistent messages put and PUT1 for this interval
 - Excellent information when trying to track down why costs may have changed!
- QQSTPPT & QQSTPP1 – Number of persistent messages put and put1 during this interval
 - Same comment!

API MQPUT and PUT1 Data - Continued!!!

- For shared queues, these numbers just reflect the API requests for this queue manager
- QQSTPUTB & QQSTPT1B – Number of bytes put to the queue including any message properties for both MQPUT and MQPUT1
 - Does not include the MQMD
- QQSTPPB & QQSTP1B – Number of persistent bytes put to the queue for PUT and PUT1
 - Also does not include Message Headers
- QQSTNPPB & QQSTNP1B – Number of nonpersistent bytes put to the queue for PUT and PUT1
 - Also does not include Message Headers
- QQSTFLPT & QQSTFLP1 – Failed put counts for both MQPUT and MQPUT1

API MQPUT and PUT1 Data - Continued!!!

- For shared queues, these numbers just reflect the API requests for this queue manager
- QQSTFPTC – Number of Fast Put to Waiting Getter MQPUTs and MQPUT1s during this interval
- QQSTFPTB – Number of bytes put to the queue via fast puts
- QQSTSTRM – Streamed Message Count
 - Always 0 if queue is not streamed
- QQSTMSMI – Smallest Message put during this interval
- QQSTMSMA – Largest Message put during the interval
- QQSTMSAV – Average message size

API MQGET Data!!!

- For shared queues, these numbers just reflect the API requests for this queue manager
- Destructive MQGET Counts for the Interval:
 - QQSTGETS – Number of Destructive MQGET requests
 - QQSTNPDG – Number of Nonpersistent Destructive MQGETs
 - QQSTPDG – Number of Persistent Destructive MQGETs
 - QQSTGETB – Total destructive bytes retrieved
 - QQSTNPDB – Total Destructive Nonpersistent bytes
 - QQSTPDB – Total Destructive Persistent bytes

API MQGET Data – Continued !!!

- For shared queues, these numbers just reflect the API requests for this queue manager
- Non-Destructive MQGET Counts (Browses) for the Interval:
 - QQSTBRWS – Number of BROWSING GET requests
 - QQSTNPBR – Number of Nonpersistent Browsing GETs
 - QQSTPBR – Number of Persistent Browsing GETs
 - QQSTBRWB – Total Browsing bytes retrieved
 - QQSTNPBB – Total Browsing Nonpersistent bytes
 - QQSTPBB – Total Browsing Persistent bytes

API MQGET Failure Counts – Not found in the WQ records

- For shared queues, these numbers just reflect the API requests for this queue manager
- MQGET Failure Counts:
 - QQSTFLGT– Failed Destructive MQGETs
 - Includes truncated message failures
 - Does not include 2033s for GETs with a Wait option
 - QQSTNMAG – Count of Failed destructive GETs that get a '2033'
 - QQSTTMFB – Count of Failed destructive GETs with a truncated message
 - QQSTFLGW – No message Available count for Destructive GETs with a wait
 - QQSTRDGW – Re-driven Destructive MQGETs with a WAIT
 - Similar fields for nondestructive gets as well

MORE COUNTs!!!

- For shared queues, these numbers just reflect the API requests for this queue manager
- QQSTIPHI – High watermark of ‘Open for Input’ connected applications
- QQSTIPLO - Low watermark of ‘Open for Input’ connected applications
- QQSTOPHI - High watermark of ‘Open for Output’ connected applications
- QQSTOPLO – Low watermark of ‘Open for Output’ connected applications

MORE COUNTs!!! – MORE STUFF!!!

- For shared queues, these numbers just reflect the API requests for this queue manager
- QQSTOPEN – Successful MQOPENs during this interval
- QQSTCLOS - Successful MQCLOSEs during this interval applications
- QQSTINQR – Number of MQINQ requests made
- QQSTSET – Number of MQSET requests
- QQSTEXPR – How many messages on this queue expired during this interval
- QQSTRBPT – Count of MQPUTs that were rolled back during the interval
- QQSTRBGT – Count of MQGETs that were rolled back during the interval

Will this replace the WQ records?

- I wish I could say YES!
- I cannot say No!
- I do say 'It Depends!'
 - There are many instances where MQ Admins, Capacity Planning staff, etc. can use trend analysis on the data
 - There are many instances when this data could provide better insight into the application code than we get from the Message Manager and Data Manager data
 - This should also help find workload balance changes that often precede performance problems.
 - We hope it will help those inheriting an MQ estate determine where there may be issues.
 - We believe it will help AI enhanced monitoring and tooling create alerts and suggest changes before problems start happening.

Cost of this collection

- 'Not significant' - according to MP16 (2022 version)
- As with the other MQ SMF data most of this data was already being collected.
- We are awaiting practical experience.

Did we get everything we asked for in QQST?

- No, but this is a lot of very helpful information!
 - At a lower cost than the accounting data.
 - And work is ongoing in this area!



A few other odds and ends

- Streaming Queues –

- The WQ records changed in MQ 9.3.2 to include information about streaming queues

<https://community.ibm.com/community/user/integration/blogs/johnny-murphy/2023/02/27/mq-for-zos-streaming-queue-accounting-data-added>

- MP1B and MQSMFCSV have been altered to include and process these fields and all the ones added in 9.3.3

Looking for Volunteers!

- If you are going to implement MQ V9.3.3 in either production or a heavy volume stress test environment
 - We would LOVE to look at this data.
 - Our test environments cannot reproduce the volumes and variety.

VOLUNTEER



Thank YOU!

