



## Monitoring and Auditing IBM MQ (z/OS & Distributed)

17898, Dolphin, Europe 3, Fri Aug 14<sup>th</sup> 2015, 08:30 – 09:30am

*Mayur Raja*

*IBM Hursley Park, Winchester, UK*

*mayur\_raja@uk.ibm.com*



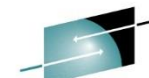
#SHAREorg



SHARE is an independent volunteer-run information technology association that provides education, professional networking and industry influence.



# MQ Sessions this week



**SHARE**  
Educate • Network • Influence

	Monday	Tuesday	Wednesday	Thursday	Friday
08:30			MQ for z/OS, Using and Abusing New Hardware and the New v8 Features	Nobody Uses Files Any More Do They? New Technologies for Old Technology, File Processing in MQ MFT and IIB	Monitoring and Auditing MQ Securing MQ Initiated CICS Workload
10:00	Introduction to MQ - Can MQ Really Make My Life Easier?	MQ for z/OS: The Insider Story	IBM Integration Bus MQ Flexibility	Common Problems and Problem Determination for MQ z/OS	IBM MQ and IBM Integration Bus - from Migration and Maintenance to Continuous Enhancements, How and Why to Stay Current
11:15	Introduction to IBM Integration Bus on z/OS	Introduction to the New MQ Appliance	MQ V8 Hands-on Labs! MQ V8 with CICS and COBOL! MQ SMF Labs!		
12:15					
1:45	What's New in the Messaging Family - MQ v8 and More		Getting Started with Performance of MQ on z/OS	IBM MQ: Are z/OS & Distributed Platforms Like Oil & Water?	
3:15	What's New in IBM Integration Bus	Live!: End to End Security of My Queue Manager on z/OS Application Programming with MQ Verbs	Digging into the MQ SMF Data	MQ Parallel Sysplex Exploitation, Getting the Best Availability from MQ on z/OS by Using Shared Queues	
4:30	MQ Security: New v8 Features Deep Dive	Live!: What's the Cloud Going to Do to My MQ Network? The Do's and Don'ts of IBM Integration Bus Performance	Giving It the Beans: Using IBM MQ as the Messaging Provider for JEE Applications in IBM WebSphere Application Server	Challenge the MQ & IIB Experts?	

Complete your session evaluations online at [www.SHARE.org/Orlando-Eval](http://www.SHARE.org/Orlando-Eval)

IBM MQ (<http://www-03.ibm.com/software/products/en/ibm-mq>)

8/3/2015

**SHARE**  
in Orlando **2015**



# Legal Disclaimer



The information contained in this publication is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this publication, it is provided AS IS without warranty of any kind, express or implied. In addition, this information is based on IBM's current product plans and strategy, which are subject to change by IBM without notice. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this publication or any other materials. Nothing contained in this publication is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

© IBM Corporation 2015. All Rights Reserved.

Complete your session evaluations online at [www.SHARE.org/Orlando-Eval](http://www.SHARE.org/Orlando-Eval)

# Please Note

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice, at IBM's sole discretion.

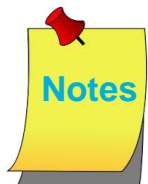
Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

# Monitoring and Auditing IBM MQ

- This session will provide illustrations and insight on the various techniques and capabilities available for MQ Administrators to meet "Monitoring" and "Auditing" demands for IBM MQ on all platforms. With the ever increasing demands to provide system runtime "high availability" along with the need to comply with the constantly changing internal and external audit demands, this session will provide information on what can be done from a IBM MQ perspective.



Complete your session evaluations online at [www.SHARE.org/Orlando-Eval](http://www.SHARE.org/Orlando-Eval)

# Different types of Monitoring

On-line  
status  
commands

Event  
messages

Off-line  
accounting  
and  
statistics

# Different types of Monitoring - Notes

- There are several different types of monitoring features in IBM MQ. You will likely use some of each type in order to look after your queue managers. Each serves a different purpose.
- There are immediate, on-line, commands that show the status of resources in the queue manager, such as queues and channels. These can be useful for diagnosing problems right now, such as why a channel is not moving messages, or why a queue has a deep current depth.
- There are event messages which the queue manager emits when certain interesting, or note-worthy occurrences happen. These can be programmatically collected, processed and acted upon.
- There are off-line statistics and accounting records cut over longer intervals accumulating information about the application activity in the queue manager which can be programmatically collected and post-processed for charge-back or capacity planning purposes. This is the one area that differs between distributed platforms and the z/OS platform.



Notes

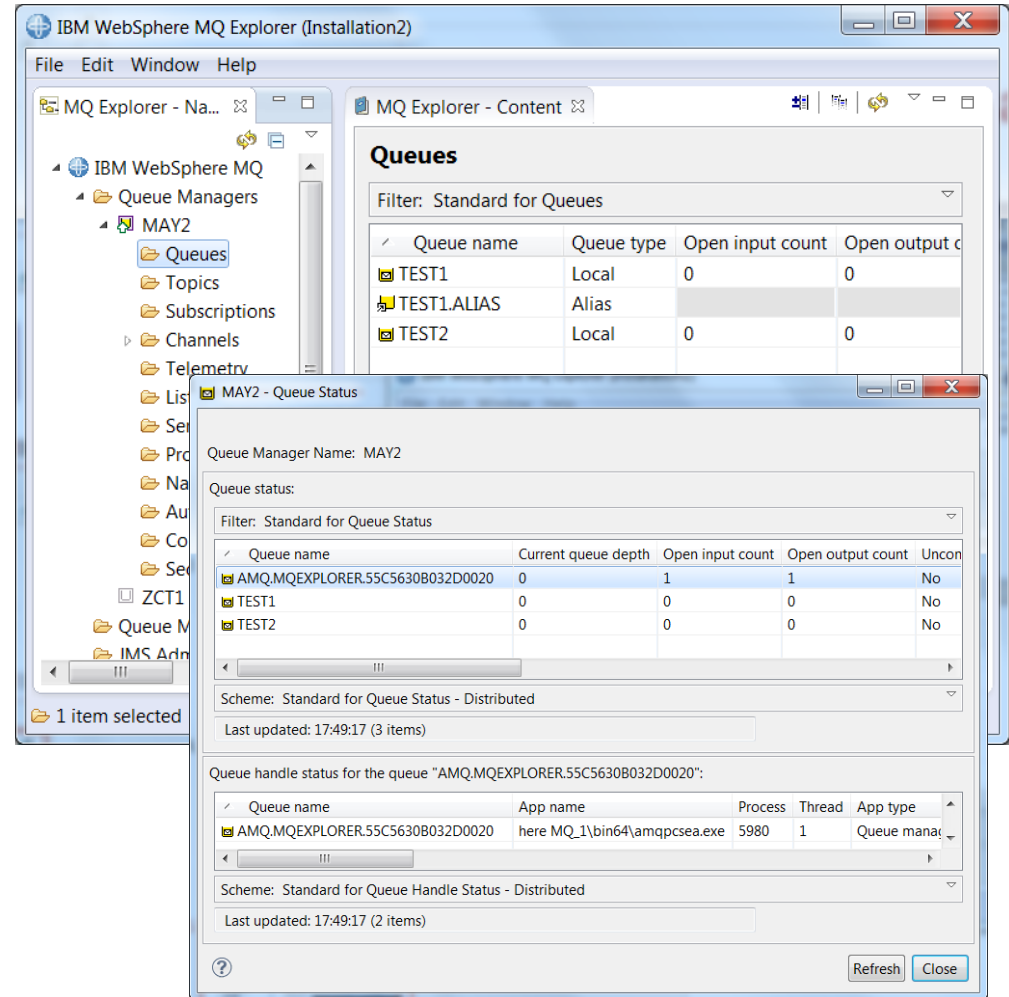
# On-line status commands

- Queue Manager
- Resources
  - Queues
  - Channels
  - Topics
- Applications
  - Connections
  - Handles
  - Subscriptions
- Available via
  - MQSC commands
  - Programmable Command Format (PCF)
  - MQ Explorer GUI

```
Command Prompt - runmqsc MAY2

C:\Program Files\IBM\WebSphere MQ_1\bin64>runmqsc MAY2
5724-H72 (C) Copyright IBM Corp. 1994, 2014.
Starting MQSC for queue manager MAY2.

dis qmstatus all
1 : dis qmstatus all
AMQ8705: Display Queue Manager Status Details.
QMNAME(MAY2)           STATUS(RUNNING)
CONNNS(22)             CMDSERU(RUNNING)
CHINIT(RUNNING)        INSTNAME(Installation2)
INSTPATH(C:\Program Files\IBM\WebSphere MQ_1)
INSTDESC( )            LDAPCONN(INACTIVE)
STANDBY(NOPERMIT)     STARTDA(2015-08-13)
STARTTI(00.04.23)
```



The screenshot shows the IBM WebSphere MQ Explorer GUI. The main window displays a tree view of the queue manager hierarchy, with 'Queues' selected under 'MAY2'. The 'Queues' panel shows a table of queues:

Queue name	Queue type	Open input count	Open output count
TEST1	Local	0	0
TEST1.ALIAS	Alias		
TEST2	Local	0	0

A secondary window titled 'MAY2 - Queue Status' is open, showing the queue status for the queue manager 'MAY2'. It displays a table of queue status:

Queue name	Current queue depth	Open input count	Open output count	Uncon
AMQ.MQEXPLORER.55C5630B032D0020	0	1	1	No
TEST1	0	0	0	No
TEST2	0	0	0	No

Below the table, it shows the scheme 'Standard for Queue Status - Distributed' and the last updated time '17:49:17 (3 items)'. A third window titled 'MAY2 - Queue Handle Status' is also open, showing the handle status for the queue 'AMQ.MQEXPLORER.55C5630B032D0020':

Queue name	App name	Process	Thread	App type
AMQ.MQEXPLORER.55C5630B032D0020	here MQ_1\bin64\amqpcsea.exe	5980	1	Queue mana

It also shows the scheme 'Standard for Queue Handle Status - Distributed' and the last updated time '17:49:17 (2 items)'. Buttons for 'Refresh' and 'Close' are visible at the bottom of the window.



# On-line status commands – Notes

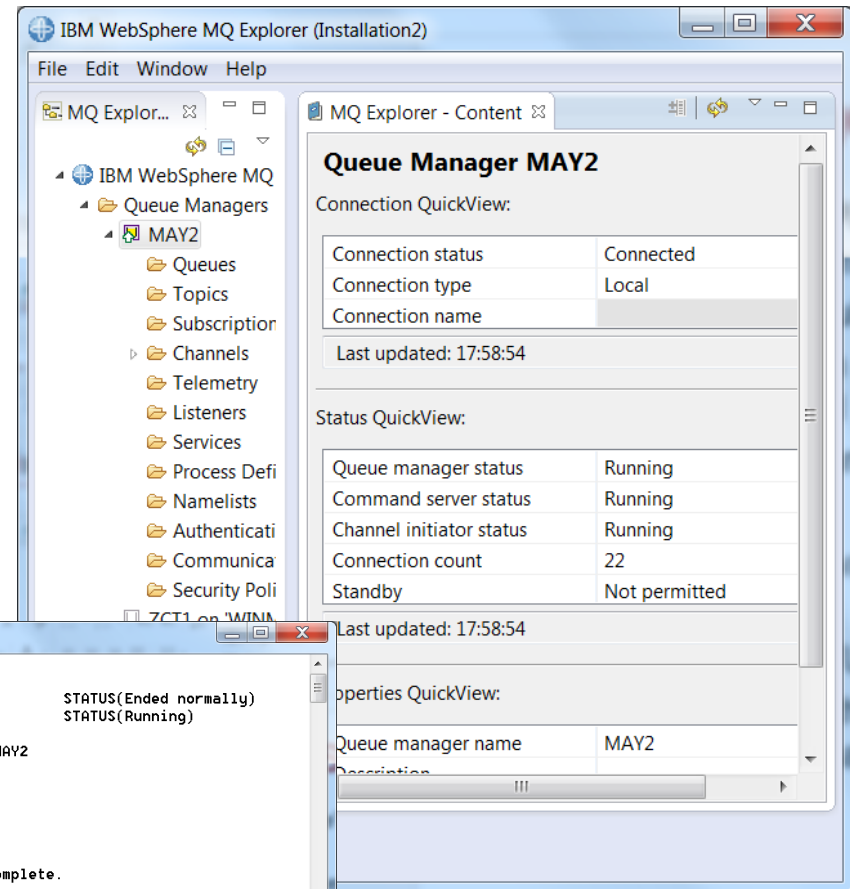
- Many of the resources in the queue manager have status commands. These commands show the run-time view of that resource.
- There is a status command for the queue manager, although being able to issue any command of course tells you something about the status of the queue manager!
- There are also status commands for channels, queues and topics, and commands which show the run-time status of applications currently connected to the queue manager, and subscriptions currently active in the system.
- Some of these status commands have additional data that you can activate. This additional data is not enabled by default because it requires the capture of time stamp information to produce and collecting time stamps can be expensive on some platforms. You enable this additional data using the MONCHL and MONQ attributes on channels and queues respectively.
- These status commands are available using MQSC commands or via PCF (and thus through the MQ Explorer GUI and other such PCF based tools). In the MQ Explorer GUI you will find status display available through context menus on a single object, or from the object folder in the Navigator (left hand pane) for status of many objects in one view.



Notes

# Queue Manager Status

- “Is the queue manager alive” ?
- ✓ PING QMGR command
- ✗ Existence of particular processes
  - Does not mean a queue manager can do anything useful
- ✓ dspmq control command
- ✗ qmstatus.ini
  - Not a reliable way to tell if a queue manager is running
- ✓ DISPLAY QMSTATUS command
  - A summary of the run-time state of the queue manager



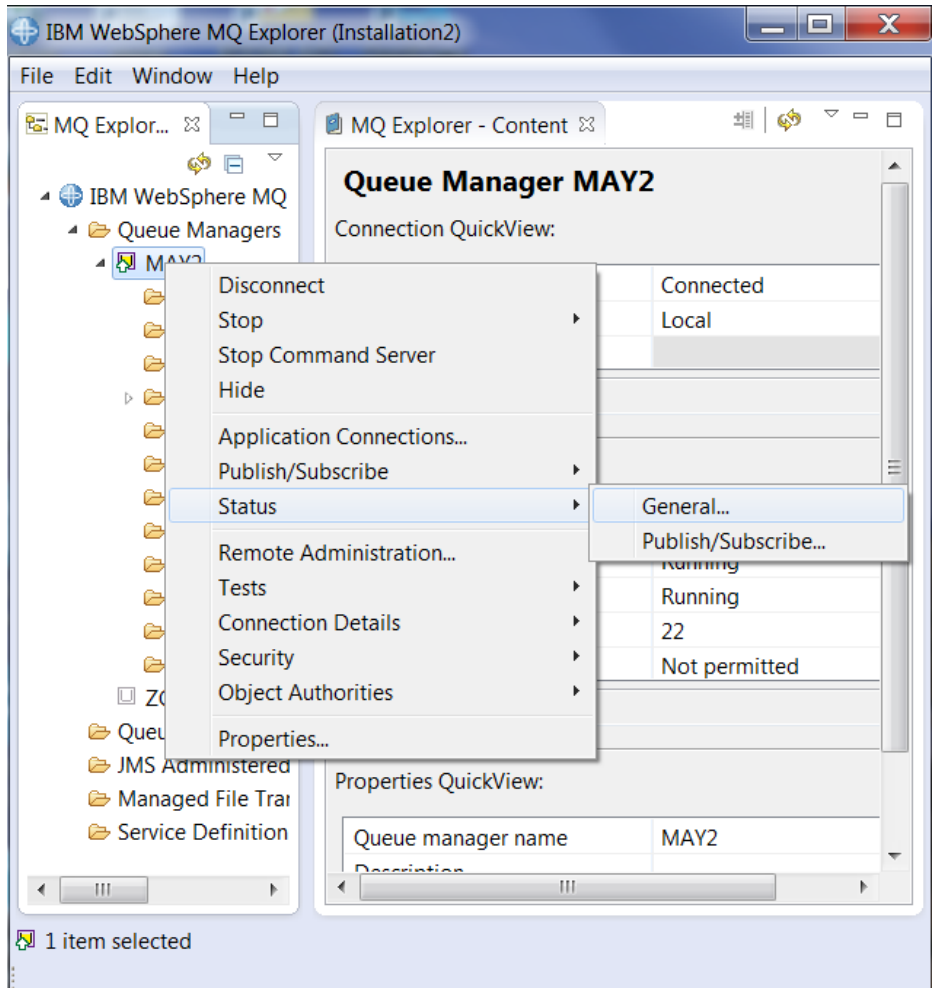
```

C:\Program Files\IBM\WebSphere MQ_1\bin64>dspmq
QMNAME(MAY1)           STATUS(Ended normally)
QMNAME(MAY2)           STATUS(Running)

C:\Program Files\IBM\WebSphere MQ_1\bin64>runmqsc MAY2
5724-H72 (C) Copyright IBM Corp. 1994, 2014.
Starting MQSC for queue manager MAY2.

ping qmgr
 1 : ping qmgr
AMQ8415: Ping WebSphere MQ Queue Manager command complete.
dis qmstatus all
 2 : dis qmstatus all
AMQ8705: Display Queue Manager Status Details.
QMNAME(MAY2)           STATUS(RUNNING)
CONNS(22)              CMDSERU(RUNNING)
CHINIT(RUNNING)        INSTNAME(Installation2)
INSTPATH(C:\Program Files\IBM\WebSphere MQ_1)
INSTDESC( )            LDAPCONN(INACTIVE)
STANDBY(NOPERMIT)     STARTDA(2015-08-13)
STARTTI(00.04.23)
  
```

# Queue Manager Status



IBM WebSphere MQ Explorer (Installation2)

File Edit Window Help

MQ Explor... MQ Explorer - Content

**Queue Manager MAY2**

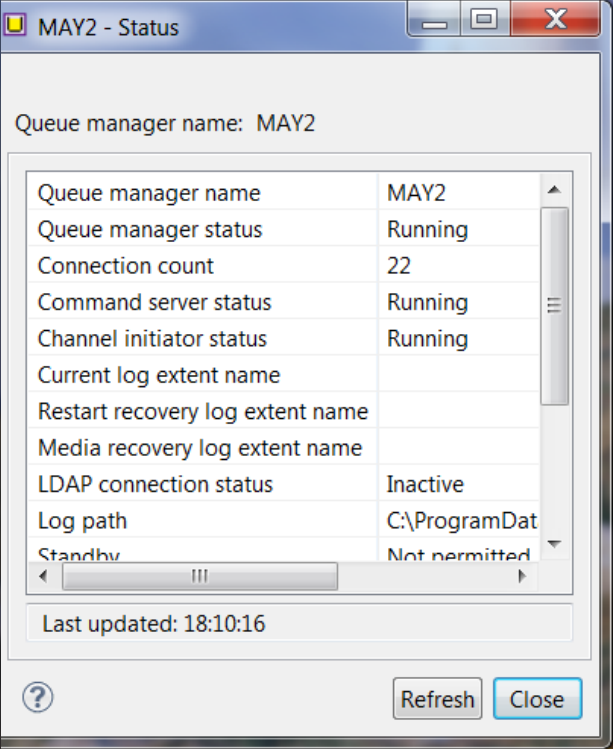
Connection QuickView:

Connected
Local

Properties QuickView:

Queue manager name	MAY2
Description	

1 item selected



MAY2 - Status

Queue manager name: MAY2

Queue manager name	MAY2
Queue manager status	Running
Connection count	22
Command server status	Running
Channel initiator status	Running
Current log extent name	
Restart recovery log extent name	
Media recovery log extent name	
LDAP connection status	Inactive
Log path	C:\ProgramDat
Standby	Not permitted

Last updated: 18:10:16

Refresh Close

# Queue Manager Status – Notes

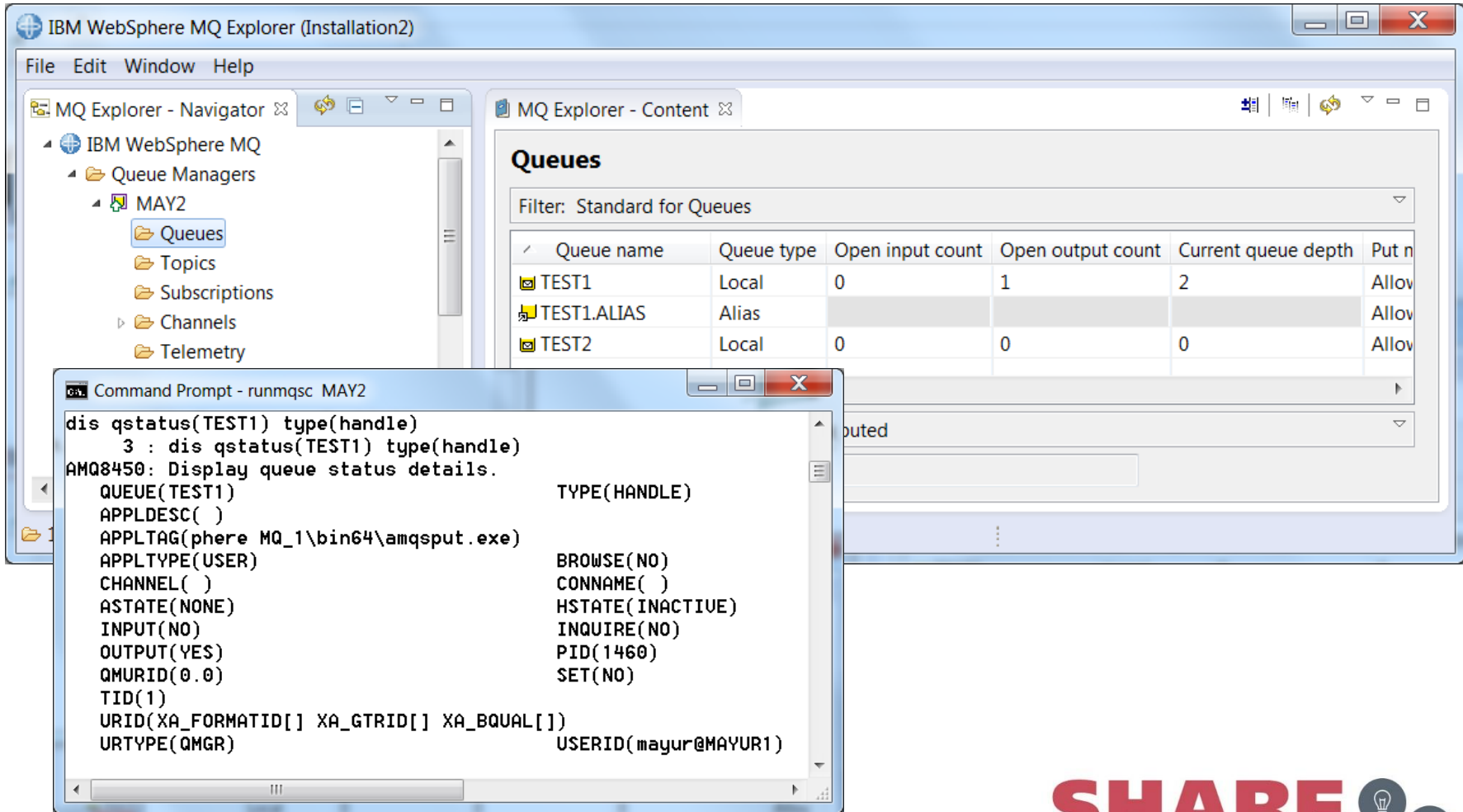
- The basic question you are trying to determine here is whether the queue manager is alive. Of course that rather depends on what you mean by “alive”. In my view it means an application can connect.
- One handy command to remember is PING QMGR. It needs several components to be working, including an application connecting, in order to complete successfully and so it is a good test of whether the queue manager is open for business. HA scripts have typically used this to determine the health of a queue manager.
- Some people want to test for the existence of particular processes. However, I don’t like this. Even if amqzma0 exists, it does not mean the queue manager can do anything useful.
- The dspmq control command is also useful as it also does an MQCONN. It is not designed to be used programmatically, also the main drawback about translated fields has been solved in V7.0.1, so it could be used in this way successfully now too.
- Some will use the contents of qmstatus.ini file. This is not a reliable method. It will correctly state if a queue manager is down, but you cannot rely on it if it says a queue manager is running, as the queue manager may have failed in a way that meant the update to this file was not able to be made.
- The status command for the queue manager, DISPLAY QMSTATUS covers a number of the run-time parts of the queue manager including the command server and channel initiator. It indicates details about the log files if you are using linear logging, and notes whether this is a multi-instance queue manager. The MQ Explorer GUI provides some of the output from this command in the quick view, and the full output can be displayed using the context menu on the queue manager and selecting Status->General...



Notes

# Status of run-time resources

- Queues
  - Run-time information about the queue
  - Details of each handle currently open against the queue



The screenshot shows the IBM WebSphere MQ Explorer interface. The left pane displays the 'Queues' folder under the 'MAY2' queue manager. The main pane shows a table of queues with the following data:

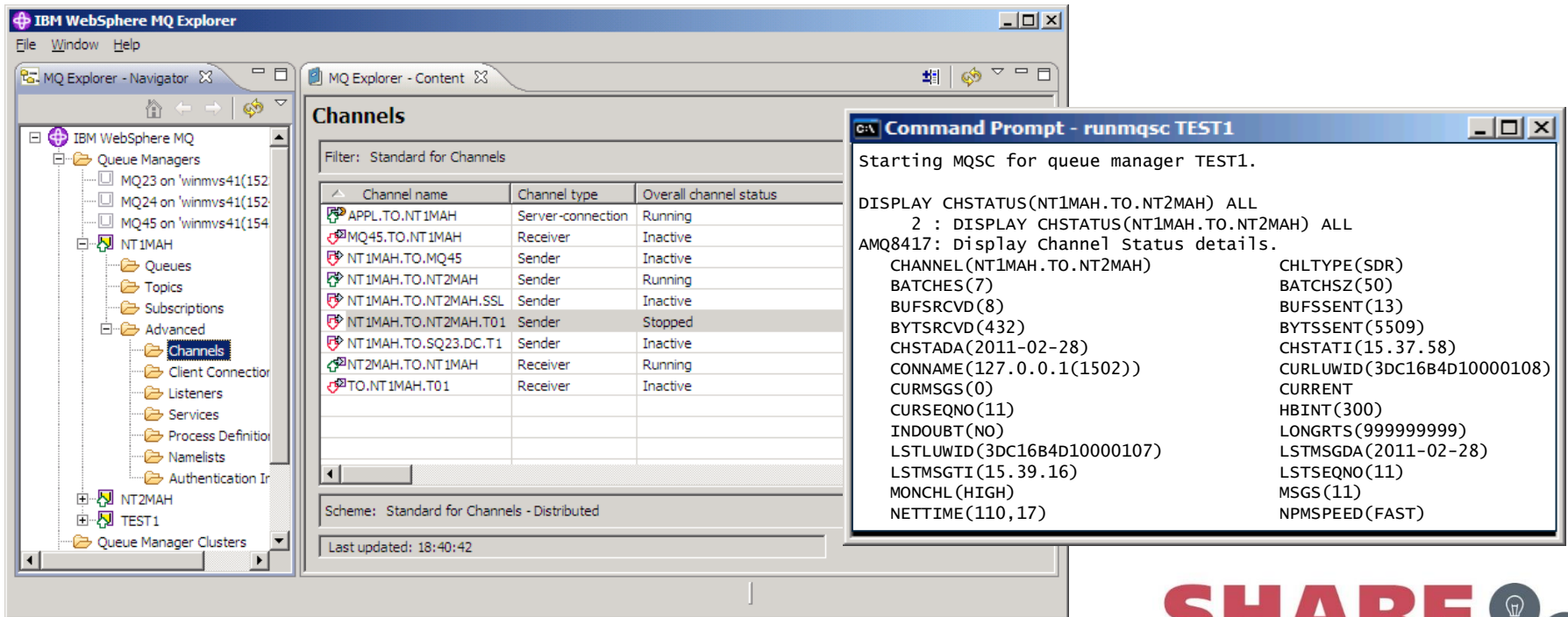
Queue name	Queue type	Open input count	Open output count	Current queue depth	Put n
TEST1	Local	0	1	2	Allow
TEST1.ALIAS	Alias				Allow
TEST2	Local	0	0	0	Allow

A Command Prompt window is overlaid on the Explorer, showing the output of the command `dis qstatus(TEST1) type(handle)`:

```
AMQ8450: Display queue status details.
QUEUE(TEST1)                                TYPE(HANDLE)
APPLDESC( )
APPLTAG(phere MQ_1\bin64\amqsput.exe)
APPLTYPE(USER)                               BROWSE(NO)
CHANNEL( )                                   CONNAME( )
ASTATE(NONE)                                 HSTATE(INACTIVE)
INPUT(NO)                                    INQUIRE(NO)
OUTPUT(YES)                                  PID(1460)
QMURID(0.0)                                  SET(NO)
TID(1)
URID(XA_FORMATID[] XA_GTRID[] XA_BQUAL[])
URTYPE(QMGR)                                USERID(mayur@MAYUR1)
```

# Status of run-time resources

- Queues
  - Run-time information about the queue
  - Details of each handle currently open against the queue
- Channels
  - Partner details
  - Logical UoW details
  - How much work it has done
  - State of channel



The screenshot shows the IBM WebSphere MQ Explorer interface. The left pane displays the 'Channels' folder under the 'NT1MAH' queue manager. The main pane shows a table of channels with their names, types, and overall statuses. A Command Prompt window is open, showing the output of the 'runmqsc TEST1' command, which displays the status of the 'NT1MAH.TO.NT2MAH' channel.

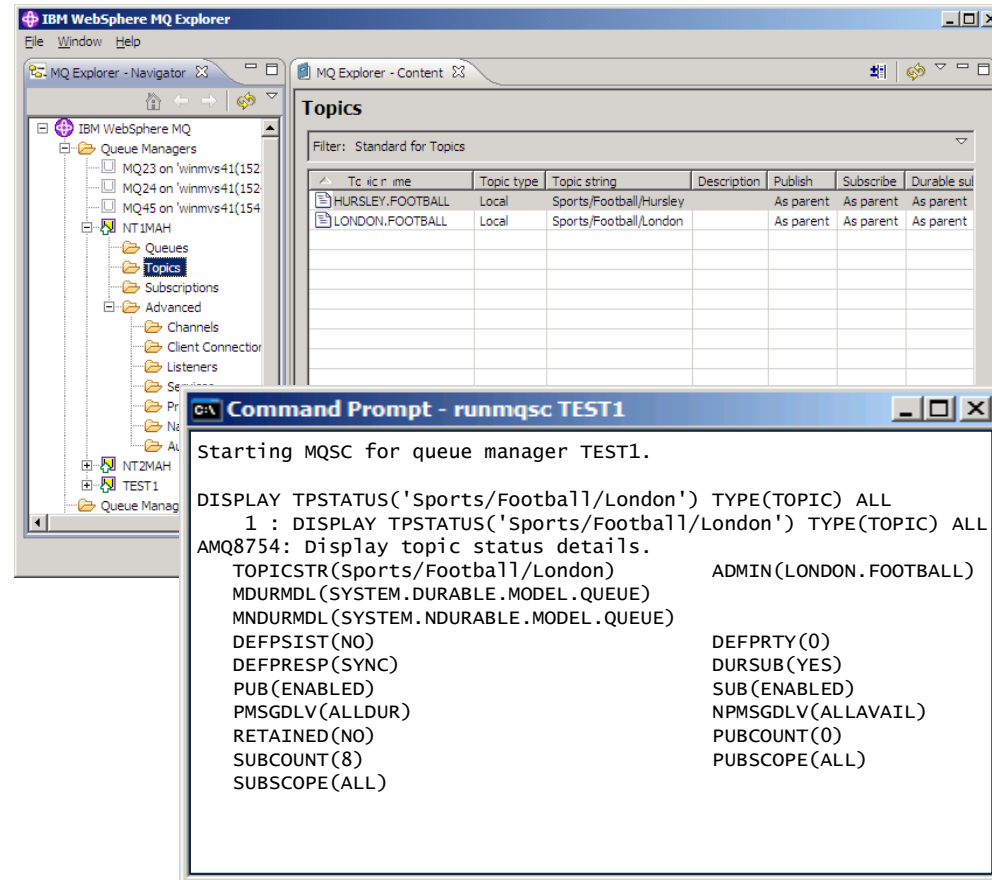
Channel name	Channel type	Overall channel status
APPL.TO.NT1MAH	Server-connection	Running
MQ45.TO.NT1MAH	Receiver	Inactive
NT1MAH.TO.MQ45	Sender	Inactive
NT1MAH.TO.NT2MAH	Sender	Running
NT1MAH.TO.NT2MAH.SSL	Sender	Inactive
NT1MAH.TO.NT2MAH.T01	Sender	Stopped
NT1MAH.TO.SQ23.DC.T1	Sender	Inactive
NT2MAH.TO.NT1MAH	Receiver	Running
TO.NT1MAH.T01	Receiver	Inactive

```
Starting MQSC for queue manager TEST1.

DISPLAY CHSTATUS(NT1MAH.TO.NT2MAH) ALL
      2 : DISPLAY CHSTATUS(NT1MAH.TO.NT2MAH) ALL
AMQ8417: Display Channel Status details.
CHANNEL(NT1MAH.TO.NT2MAH)           CHLTYPE(SDR)
BATCHES(7)                          BATCHSZ(50)
BUFRCVD(8)                          BUFSSENT(13)
BYTRCVD(432)                        BYTSSENT(5509)
CHSTADA(2011-02-28)                  CHSTATI(15.37.58)
CONNAME(127.0.0.1(1502))             CURLUWID(3DC16B4D1000108)
CURMSG(0)                            CURRENT
CURSEQNO(11)                         HBINT(300)
INDOUBT(NO)                          LONGRTS(999999999)
LSTLUWID(3DC16B4D1000107)            LSTMSGDA(2011-02-28)
LSTMSGTI(15.39.16)                  LSTSEQNO(11)
MONCHL(HIGH)                         MSGS(11)
NETTIME(110,17)                      NPMSPEED(FAST)
```

# Status of run-time resources

- Queues
  - Run-time information about the queue
  - Details of each handle currently open on the queue
- Channels
  - Partner details
  - Logical UoW details
  - How much work it has done
  - State of channel
- Topics
  - Resolution of hierarchically inherited attributes
  - Publisher Status
  - Subscriber Status



The screenshot shows the IBM WebSphere MQ Explorer interface. The left pane displays a tree view of the MQ environment, including Queue Managers, Queues, Topics, Subscriptions, and Channels. The right pane shows the 'Topics' view with a table of topic details. Below the Explorer, a Command Prompt window titled 'runmqsc TEST1' displays the output of the 'DISPLAY TPSTATUS' command for the topic 'Sports/Football/London'.

Topic name	Topic type	Topic string	Description	Publish	Subscribe	Durable sub
HURSLEY.FOOTBALL	Local	Sports/Football/Hursley		As parent	As parent	As parent
LONDON.FOOTBALL	Local	Sports/Football/London		As parent	As parent	As parent

```
Starting MQSC for queue manager TEST1.

DISPLAY TPSTATUS('Sports/Football/London') TYPE(TOPIC) ALL
1 : DISPLAY TPSTATUS('Sports/Football/London') TYPE(TOPIC) ALL
AMQ8754: Display topic status details.
TOPICSTR(Sports/Football/London)          ADMIN(LONDON.FOOTBALL)
MDURMDL(SYSTEM.DURABLE.MODEL.QUEUE)
MNDURMDL(SYSTEM.NDURABLE.MODEL.QUEUE)
DEFPERSIST(NO)                            DEFPRTY(0)
DEFPRESP(SYNC)                             DURSUB(YES)
PUB(ENABLED)                               SUB(ENABLED)
PMSGDLV(ALLDUR)                            NPMSGDLV(ALLAVAIL)
RETAINED(NO)                               PUBCOUNT(0)
SUBCOUNT(8)                               PUBSCOPE(ALL)
SUBSCOPE(ALL)
```

# Status of run-time resources – Notes

- There are status commands allowing the run-time view of various IBM MQ resources to be queried.
- For queues, there is DISPLAY QSTATUS command which has two variants. DISPLAY QSTATUS TYPE(Queue) shows overall information about the queue, for example, how many applications have an input or output handle open on that queue, the depth of the queue, whether there is any uncommitted work on this queue and when was the last time the queue was processed either to get or put messages. A second variant, DISPLAY QSTATUS TYPE(HANDLE) shows more detailed information about each handle that currently has the queue open, including information about the application that owns the handle.
- For channels, there is DISPLAY CHSTATUS command which shows the run-time information for each channel instance – remembering of course that there may be multiple channel instances of the same name in the case of receiver and server-connection channels. The information displayed includes the details of the partner we are connected to; any logical unit of work information in the case of queue manager to queue manager channels; how much work the channel has done; and of course the state of the channel.
- For topics, there is DISPLAY TPSTATUS command which has three variants. DISPLAY TPSTATUS TYPE(TOPIC) shows the resolution of any ASPARENT definitional values. Topics are defined in a hierarchical structure which we call the topic tree. Any values which are not specifically set on a topic are inherited from the parent in the topic tree. This command makes the actual run-time values easy to see without having to display many different topic objects. Two further variants, DISPLAY TPSTATUS TYPE(PUB) and TYPE(SUB) show more detailed information about each handle that currently has the topic open, including the connection ID which can be used with the DISPLAY CONN command to discover full information about the application that owns the handle.



Notes



# Command Examples: Queues and Topics

```
DISPLAY QSTATUS(Q1) TYPE(Queue)
1 : DISPLAY QSTATUS(Q1) TYPE(Queue)
AMQ8450: Display queue status details.
```

```
Queue(Q1)                               TYPE(Queue)
CURDEPTH(5)                             IPPROCS(0)
LGETDATE(2011-02-28)                    LGETTIME(14.58.36)
LPUTDATE(2011-02-28)                    LPUTTIME(14.58.52)
MEDIALOG(S0000000.LOG)                  MONQ(HIGH)
MSGAGE(29)                              OPPROCS(1)
QTIME(47076388, 45807040)              UNCOM(NO)
```

```
DISPLAY QSTATUS(Q1) TYPE(Handle)
2 : DISPLAY QSTATUS(Q1) TYPE(Handle)
AMQ8450: Display queue status details.
```

```
Queue(Q1)                               TYPE(Handle)
APPLDESC( )                             APPLTAG(d:\nttools\q.exe)
APPLTYPE(USER)                          BROWSE(NO)
CHANNEL(SYSTEM.DEF.SVRCONN)             CONNAME(127.0.0.1)
ASTATE(NONE)                             HSTATE(INACTIVE)
INPUT(NO)                                INQUIRE(NO)
OUTPUT(YES)                              PID(14608)
QMURID(0.0)                              SET(NO)
TID(*)
URID(XA_FORMATID[00000000] XA_GTRID[] XA_BQUAL[])
URTYPE(QMGR)                             USERID(hughson@R820DGM)
```

```
DISPLAY TPSTATUS('Sports/Football/London') TYPE(Topic) ALL
1 : DISPLAY TPSTATUS('Sports/Football/London') TYPE(Topic) ALL
AMQ8754: Display topic status details.
```

```
TOPICSTR(Sports/Football/London)        ADMIN(LONDON.FOOTBALL)
MDURMDL(SYSTEM.DURABLE.MODEL.Queue)    DURSUB(YES)
MNDURMDL(SYSTEM.NDURABLE.MODEL.Queue)   DEFPRTY(0)
DEFPSIST(NO)                            DEFRESP(SYNC)
DEFPRES(PRES)                            PUB(ENABLED)
PUB(ENABLED)                             NPMSGDLV(ALLAVAIL)
PMSGDLV(ALLDUR)                          PUBCOUNT(0)
RETAINED(NO)                              PUBSCOPE(ALL)
SUBCOUNT(8)
SUBSCOPE(ALL)
```

```
DISPLAY TPSTATUS('Sports/Football/London') TYPE(Pub) ALL
2 : DISPLAY TPSTATUS('Sports/Football/London') TYPE(Pub) ALL
AMQ8754: Display topic status details.
```

```
TOPICSTR(Sports/Football/London)        LPUBDATE(2011-02-28)
LPUBTIME(16:11:50)
ACTCONN(414D51434E54314D41482020202020203FC16B4D20002401)
NUMPUBS(1)
```

```
DISPLAY TPSTATUS('Sports/Football/London') TYPE(Sub) ALL
3 : DISPLAY TPSTATUS('Sports/Football/London') TYPE(Sub) ALL
AMQ8754: Display topic status details.
```

```
TOPICSTR(Sports/Football/London)
SUBID(414D51204E54314D41482020202020203FC16B4D2000280A)
SUBUSER(hughson)                          RESMDATE(2011-02-28)
RESMTIME(16:12:49)                        LMSGDATE(2011-02-28)
LMSGTIME(16:12:51)
ACTCONN(414D51434E54314D41482020202020203FC16B4D20002807)
DURABLE(NO)                               SUBTYPE(API)
NUMMSG(4)
```



Notes

# Command Examples: Channels

```
DISPLAY CHSTATUS(SYSTEM.DEF.SVRCONN) ALL
```

```
1 : DISPLAY CHSTATUS(SYSTEM.DEF.SVRCONN) ALL
```

```
AMQ8417: Display Channel Status details.
```

```
CHANNEL(SYSTEM.DEF.SVRCONN)      CHLTYPE(SVRCONN)
BUFSRCVD(9)                       BUFSSENT(8)
BYTSRCVD(1688)                    BYTSSENT(1516)
CHSTADA(2011-02-28)               CHSTATI(15.30.53)
COMPHDR(NONE,NONE)                COMPMSG(NONE,NONE)
COMPRATE(0,0)                     COMPTIME(0,0)
CONNAME(127.0.0.1)                CURRENT
EXITTIME(0,0)                     HBINT(300)
JOBNAME(0000391000003B14)          LOCLADDR( )
LSTMSGDA(2011-02-28)              LSTMSGTI(15.30.53)
MCASTAT(RUNNING)                  MCAUSER(hughson)
MONCHL(OFF)                        MSGS(5)
RAPPLTAG(d:\nttools\q.exe)        SSLCERTI( )
SSLKEYDA( )                       SSLKEYTI( )
SSLPEER( )                        SSLRKEYS(0)
STATUS(RUNNING)                   STOPREQ(NO)
SUBSTATE(RECEIVE)                 CURSHCNV(1)
MAXSHCNV(10)
```

```
DISPLAY CHSTATUS(NT1MAH.TO.NT2MAH) ALL
```

```
2 : DISPLAY CHSTATUS(NT1MAH.TO.NT2MAH) ALL
```

```
AMQ8417: Display Channel Status details.
```

```
CHANNEL(NT1MAH.TO.NT2MAH)        CHLTYPE(SDR)
BATCHES(7)                        BATCHSZ(50)
BUFSRCVD(8)                       BUFSSENT(13)
BYTSRCVD(432)                     BYTSSENT(5509)
CHSTADA(2011-02-28)               CHSTATI(15.37.58)
COMPHDR(NONE,NONE)                COMPMSG(NONE,NONE)
COMPRATE(0,0)                     COMPTIME(0,0)
CONNAME(127.0.0.1(1502))          CURLUID(3DC16B4D10000108)
CURMSGS(0)                         CURRENT
CURSEQNO(11)                       EXITTIME(0,0)
HBINT(300)                         INDOUBT(NO)
JOBNAME(0000397800003634)          LOCLADDR(127.0.0.1(1901))
LONGRTS(999999999)                LSTLUWID(3DC16B4D10000107)
LSTMSGDA(2011-02-28)              LSTMSGTI(15.39.16)
LSTSEQNO(11)                       MCASTAT(RUNNING)
MONCHL(HIGH)                       MSGS(11)
NETTIME(110,17)                   NPMSPEED(FAST)
RQMNAME(NT2MAH)                   SHORTRTS(10)
SSLCERTI( )                       SSLKEYDA( )
SSLKEYTI( )                        SSLPEER( )
SSLRKEYS(0)                        STATUS(RUNNING)
STOPREQ(NO)                        SUBSTATE(MQGET)
XBATCHSZ(1,1)                      XMITQ(NT2MAH)
XQTIME(1794,344)
```



Notes

Complete your session evaluations online at [www.SHARE.org/Orlando-Eval](http://www.SHARE.org/Orlando-Eval)

# Status of Application Connections

- Application connection
  - What
  - Who
  - UoW details
- Application resources
  - Handles opened on queues, topics and subscriptions

**NT1MAH - Application Connections**

Applications connected to "NT1MAH":

App name	App type	App description	Process	Thread	User ID	Options	Channel name	Co
C:\mqm\java\jre\bin\javaw.exe	User		24128	9	hughson	Isolated, Share block		
C:\mqm\bin\amqpcsea.exe	Queue manager	WebSphere MQ Command Server	16368	1	MUSR_MQADMIN	Shared		
C:\mqm\bin\amqrmppa.exe	Queue manager	WebSphere MQ Channel	26180	3	MUSR_MQADMIN	Shared		
C:\mqm\bin\amqrmfma.exe	Queue manager	WebSphere MQ Cluster Repository	18560	1	MUSR_MQADMIN	Shared		
C:\mqm\bin\amqzdmaa.exe	Queue manager	WebSphere MQ Deferred Message Processor	20120	1	MUSR_MQADMIN	Shared		
C:\mqm\bin\amqzfuma.exe	Queue manager	WebSphere MQ Object Authority Manager	21356	1	MUSR_MQADMIN	Fastpath		
C:\mqm\bin\runmqchl.exe	Queue manager	WebSphere MQ Channel Initiator	14972	1	MUSR_MQADMIN	Shared		
C:\mqm\bin\runmqchl.exe	Queue manager	WebSphere MQ Channel	26452	1	MUSR_MQADMIN	Shared		
d:\nttools\q.exe	User		24960	1	hughson	Shared		

Scheme: Standard for Application Connections - Distributed  
Last updated: 19:46:19

Queue manager objects accessed by application "d:\nttools\q.exe":

Object name	Object type	Open options	Topic string	Subscription name	Subscription ID	Destination queue manager
Q1	Queue	Fail if quiescing, Input shared				
Q2	Queue	Output, Fail if quiescing				

Scheme: Standard for Application Connection Handles - Distributed  
Last updated: 19:46:19

Refresh Close

**Command Prompt - runmqsc TEST1**

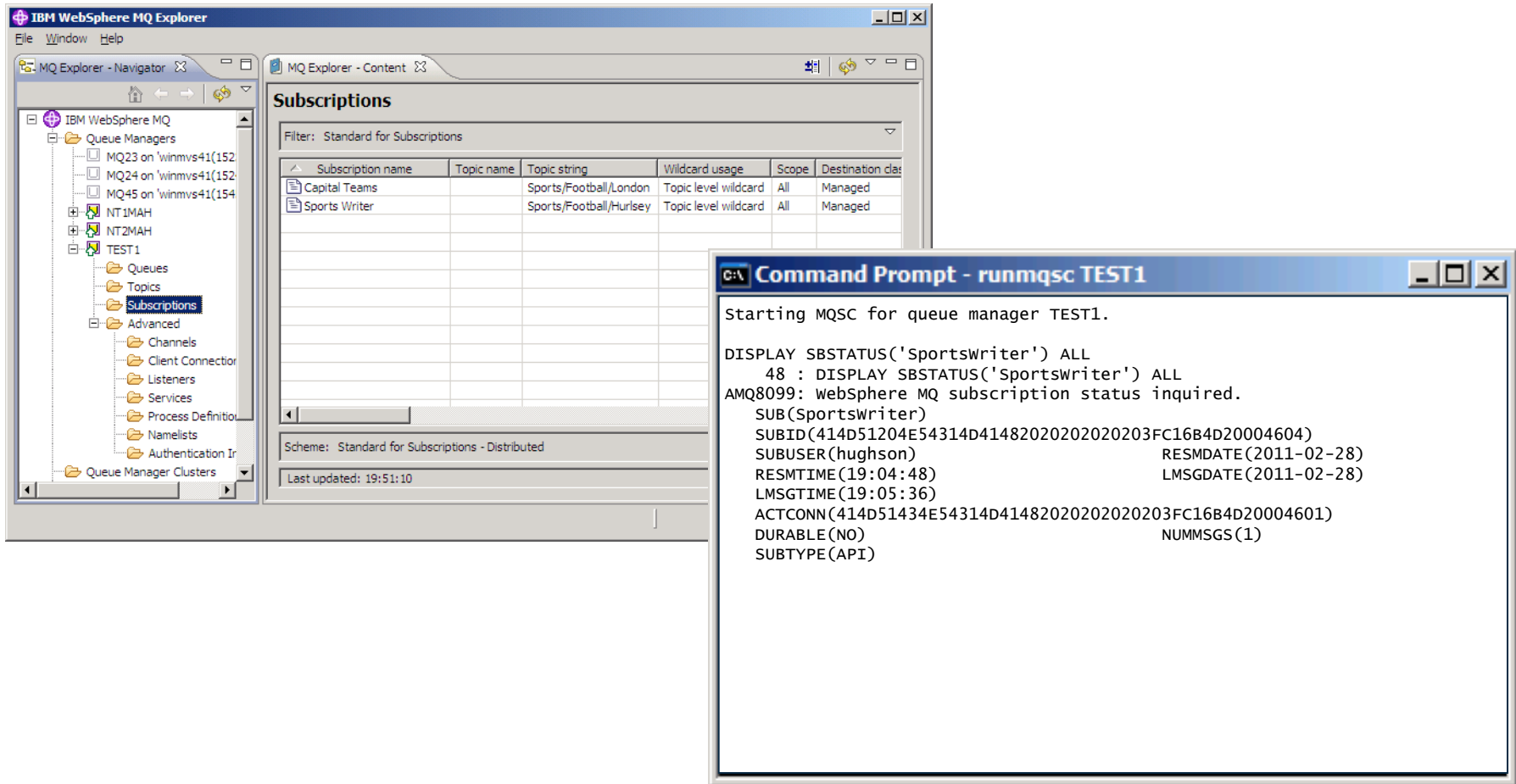
```
Starting MQSC for queue manager TEST1.

DISPLAY CONN(*) WHERE(APPLTYPE EQ USER) TYPE(ALL) ALL
1 : DISPLAY CONN(*) WHERE(APPLTYPE EQ USER) TYPE(ALL) ALL
AMQ8276: Display connection details.
CONN(3FC16B4D20002461)
EXTCONN(414D51434E54314D4148202020202020)
TYPE(CONN)
APPLTAG(d:\nttools\q.exe)
APPLTYPE(USER)
CONNOPTS(MQCNO_SHARED_BINDING)
ASTATE(NONE)
USERID(hughson)

OBJNAME( )
OBJTYPE(TOPIC)
ASTATE(NONE)
HSTATE(INACTIVE)
OPENOPTS(MQOO_OUTPUT,MQOO_FAIL_IF QUIESCING)
TOPICSTR(Sports/Football/London)
```

# Status of Subscriptions – owner ?

- Subscription definitions
  - Subscription owner, subscription type



The screenshot shows the IBM WebSphere MQ Explorer interface. On the left, a tree view displays the hierarchy of the queue manager, including Queue Managers, Queues, Topics, Subscriptions, and Advanced. The 'Subscriptions' folder is selected. The main pane shows a table of subscriptions with the following data:

Subscription name	Topic name	Topic string	Wildcard usage	Scope	Destination class
Capital Teams		Sports/Football/London	Topic level wildcard	All	Managed
Sports Writer		Sports/Football/Hurley	Topic level wildcard	All	Managed

Below the table, the scheme is identified as 'Standard for Subscriptions - Distributed' and the last updated time is '19:51:10'.

Overlaid on the bottom right is a Command Prompt window titled 'C:\> Command Prompt - runmqsc TEST1'. It displays the following output:

```
Starting MQSC for queue manager TEST1.

DISPLAY SBSTATUS('Sportswriter') ALL
48 : DISPLAY SBSTATUS('Sportswriter') ALL
AMQ8099: WebSphere MQ subscription status inquired.
SUB(Sportswriter)
SUBID(414D51204E54314D41482020202020203FC16B4D20004604)
SUBUSER(hughson) RESMDATE(2011-02-28)
RESMTIME(19:04:48) LMSGDATE(2011-02-28)
LMSGTIME(19:05:36)
ACTCONN(414D51434E54314D41482020202020203FC16B4D20004601)
DURABLE(NO) NUMMSG(1)
SUBTYPE(API)
```

# Status of Applications – Notes

- DISPLAY CONN provides information about the applications connected to the queue manager and the handles that they have open. Various other commands output a connection ID which can be used as input into DISPLAY CONN in order to find out full details about the application that made the connection in question. You can find out what the application is, whether client or locally connected for example; the user id it is running under; and when the last UoW was started – helpful for diagnosing long running UoWs.
- You can also see details of the handles this connection has open. This shows the same information as DISPLAY QSTATUS and DISPLAY TPSTATUS TYPE(PUB/SUB) but in a different perspective. Those other commands looked at the information from the perspective of the resource and who is using the resources. This command shows the information in the perspective of the application and what resources it is using. It is useful to know that you can start from either perspective and get the same information.
- DISPLAY SBSTATUS provides the same information about subscriptions as DISPLAY TPSTATUS TYPE(SUB) but from the perspective of a subscription rather than the resource, as one subscription may cover a number of topics if it has been made using a wildcarded string.



Notes

Complete your session evaluations online at [www.SHARE.org/Orlando-Eval](http://www.SHARE.org/Orlando-Eval)

# Command Examples: Connections

```
DISPLAY CONN(*) WHERE(APPLTYPE EQ USER) TYPE(ALL) ALL
  1 : DISPLAY CONN(*) WHERE(APPLTYPE EQ USER) TYPE(ALL) ALL
AMQ8276: Display Connection details.
CONN(3FC16B4D20002461)
EXTCONN(414D51434E54314D4148202020202020)
TYPE(CONN)
PID(16932)                TID(1)
APPLDESC( )              APPLTAG(d:\nttools\q.exe)
APPLTYPE(USER)          ASTATE(NONE)
CHANNEL( )              CONNAME( )
CONNOPTS(MQCNO_SHARED_BINDING)  USERID(hughson)
UOWLOG( )              UOWSTDA( )
UOWSTTI( )            UOWLOGDA( )
UOWLOGTI( )           URTYPE(QMGR)
EXTURID(XA_FORMATID[00000000] XA_GTRID[] XA_BQUAL[])
QMURID(0.0)            UOWSTATE(NONE)

OBJNAME( )              OBJTYPE(TOPIC)
ASTATE(NONE)           HSTATE(INACTIVE)
OPENOPTS(MQOO_OUTPUT,MQOO_FAIL_IF QUIESCING)
READA(NO)
TOPICSTR(Sports/Football/London)
```

```
DISPLAY CONN(*) WHERE(APPLTYPE EQ USER) TYPE(ALL) ALL
  1 : DISPLAY CONN(*) WHERE(APPLTYPE EQ USER) TYPE(ALL) ALL
AMQ8276: Display Connection details.
CONN(3FC16B4D20003B01)
EXTCONN(414D51434E54314D4148202020202020)
TYPE(CONN)
PID(17740)                TID(3)
APPLDESC( )              APPLTAG(d:\nttools\q.exe)
APPLTYPE(USER)          ASTATE(STARTED)
CHANNEL(SYSTEM.DEF.SVRCONN)  CONNAME(127.0.0.1)
CONNOPTS(MQCNO_HANDLE_SHARE_BLOCK,MQCNO_SHARED_BINDING)
USERID(hughson)        UOWLOG( )
UOWSTDA(2011-02-28)    UOWSTTI(18.13.53)
UOWLOGDA( )           UOWLOGTI( )
URTYPE(QMGR)
EXTURID(XA_FORMATID[00000000] XA_GTRID[] XA_BQUAL[])
QMURID(0.57)          UOWSTATE(ACTIVE)

OBJNAME(Q1)            OBJTYPE(QUEUE)
ASTATE(ACTIVE)        HSTATE(ACTIVE)
OPENOPTS(MQOO_INPUT_SHARED,MQOO_FAIL_IF QUIESCING)
READA(NO)
```



Notes

Complete your session evaluations online at [www.SHARE.org/Orlando-Eval](http://www.SHARE.org/Orlando-Eval)

# Command Examples: Subscriptions

```
DISPLAY SBSTATUS('sportswriter') ALL  
  48 : DISPLAY SBSTATUS('sportswriter') ALL  
AMQ8099: webSphere MQ subscription status inquired.  
SUB(sportswriter)  
SUBID(414D51204E54314D41482020202020203FC16B4D20004604)  
SUBUSER(hughson)                RESMDATE(2011-02-28)  
RESMTIME(19:04:48)              LMSGDATE(2011-02-28)  
LMSGTIME(19:05:36)  
ACTCONN(414D51434E54314D41482020202020203FC16B4D20004601)  
DURABLE(NO)                      NUMMSGS(1)  
SUBTYPE(API)
```



Notes

Complete your session evaluations online at [www.SHARE.org/Orlando-Eval](http://www.SHARE.org/Orlando-Eval)

# IBM MQ Event Messages

Complete your session evaluations online at [www.SHARE.org/Orlando-Eval](http://www.SHARE.org/Orlando-Eval)

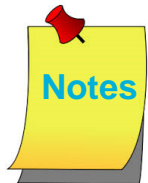


# IBM MQ Event Messages

- Event Messages for Auditing
  - Security Failures
  - Commands Issued
  - Configuration Changes
- Event Messages for Monitoring
  - Starts and Stops of resources
  - Starts and Stops of Channels
  - Channel errors
  - Application errors using resources
  - Performance of message processing
    - Servicing of queues
    - Depth of queues

# IBM MQ Event Messages – Notes

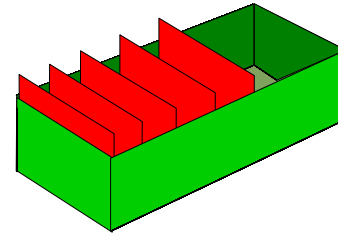
- When IBM MQ needs to emit information it often does so in the form of event messages. Several of the auditing features in IBM MQ make use of this mechanism as well as many monitoring features so we will take a general look at event messages and how they are built and then we will look at the details for the various features that use event messages.



Complete your session evaluations online at [www.SHARE.org/Orlando-Eval](http://www.SHARE.org/Orlando-Eval)

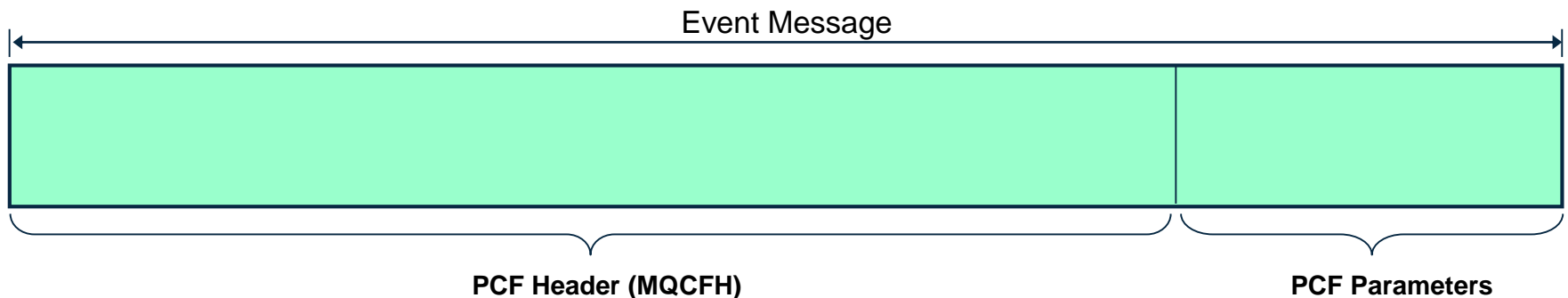
# Event Messages

- Written to specific named queues
  - Created as LOCAL queues in the default set of objects
  - Can be redefined
- Controlled by ALTER QMGR switch
- Message Descriptor
  - Format = MQFMT\_EVENT
  - 'EVENT '
- Well defined message format
  - MQCFH header
  - PCF sub-structures



SYSTEM.ADMIN.<feature name>.EVENT

**ALTER QMGR <type>EV(ENABLED)**



# Event Messages - Notes

- Event messages are used for various features of MQ and they share a common mechanism for configuration and format for the emitted events.
- As we look at each specific event feature later we will see each event type is written to a specific queue name which follows the pattern `SYSTEM.ADMIN.<feature name>.EVENT`, and is controlled by a switch, an attribute on `ALTER QMGR`, usually with `ENABLED/DISABLED` as the values (although sometimes there are additional options). The switches also follow a pattern of `<type>EV` as the name.
- The format of event messages is shown as `MQFMT_EVENT` ('EVENT ') in the message descriptor (MQMD) which is a PCF header. This means the message starts with a header using the structure `MQCFH`. This header is then followed by 1 or more parameter structures which are self-describing sub-structures providing the data of the actual event message.

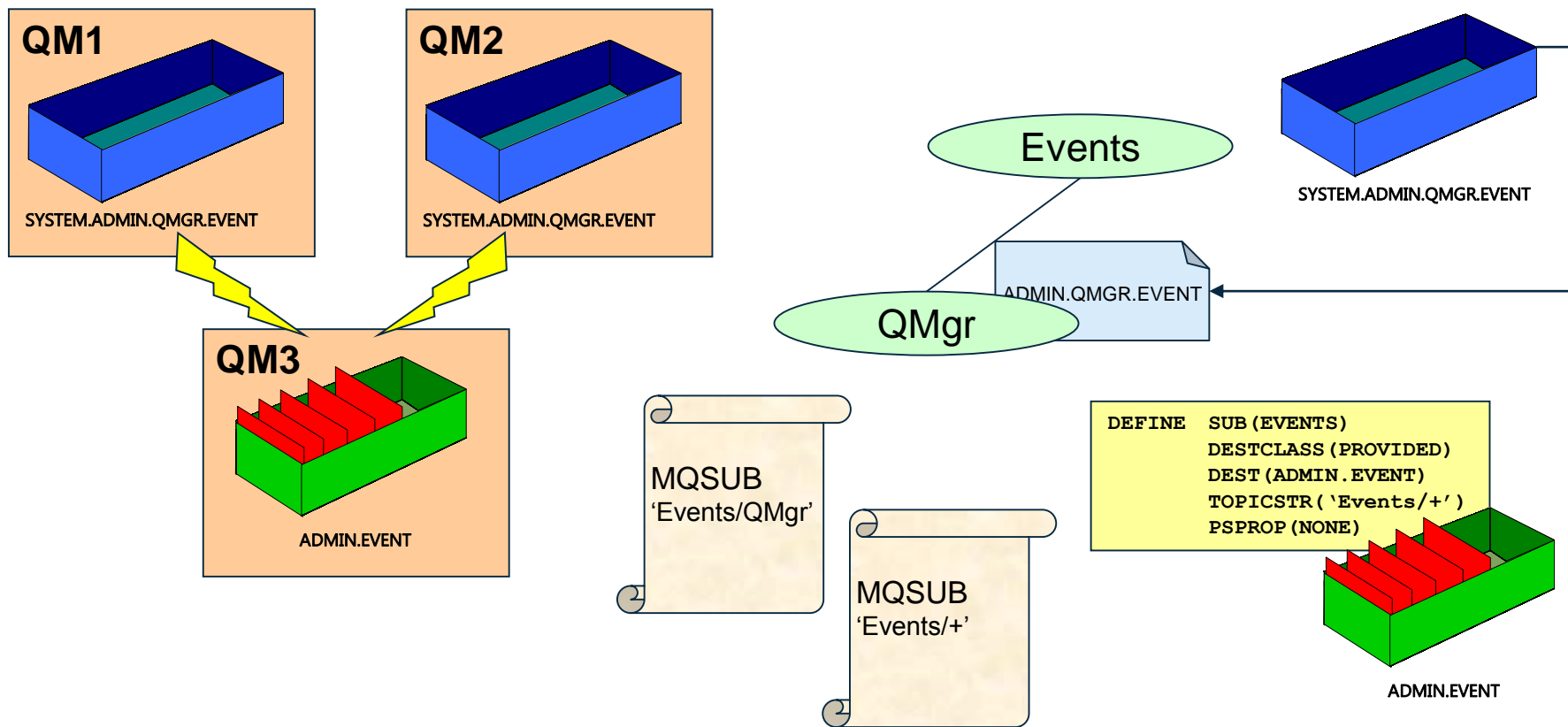


Notes

Complete your session evaluations online at [www.SHARE.org/Orlando-Eval](http://www.SHARE.org/Orlando-Eval)

# Redefining Event Queues

- Can be redefined as QREMOTE queues
- Can be redefined as QALIAS -> TOPIC



# Redefining Event Queues - Notes

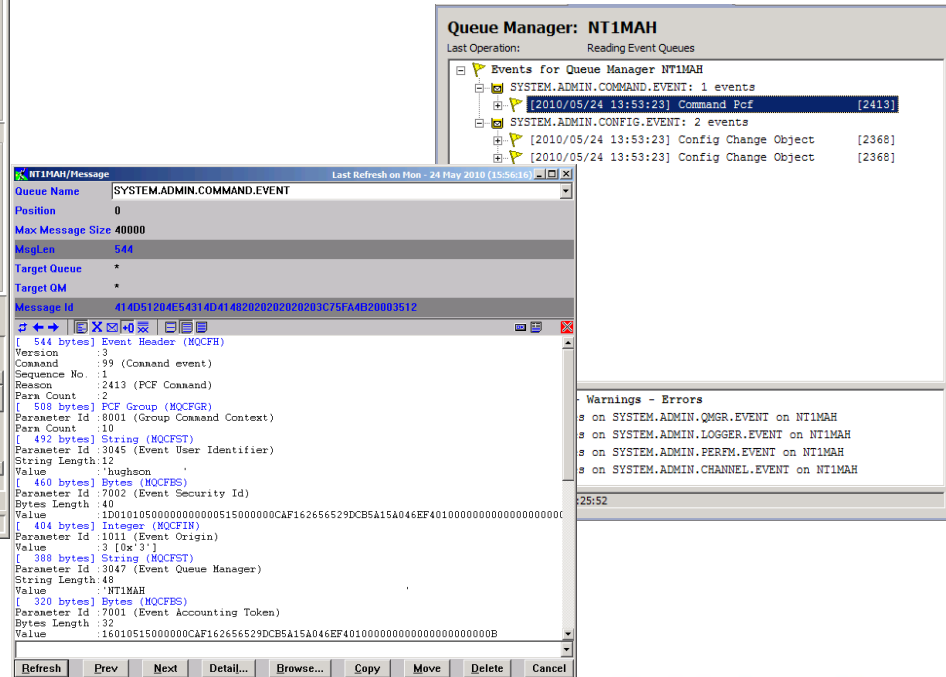
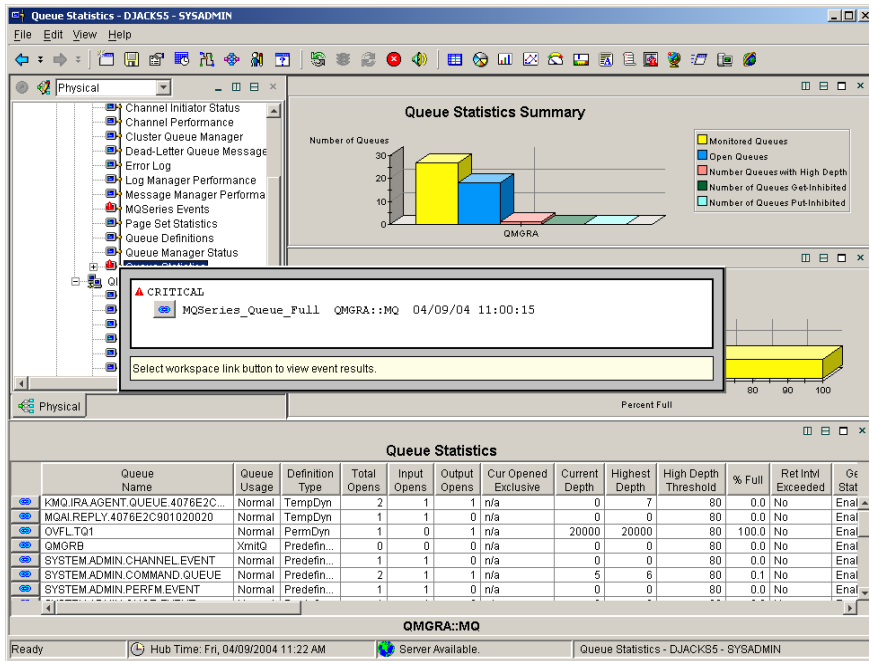
- The various event queues are defined as part of the default objects on the queue manager (on distributed platforms) and in CSQ4INSG CSQINP2 sample (on z/OS) as LOCAL queues. They can be redefined as REMOTE queues to funnel all event messages to a central queue manager to process if wished; or redefined as ALIAS queues pointing at a TOPIC object to cause them to be published. The queue name just needs to be correct.
- In doing this you can also of course funnel the events messages onto a single queue to have all event types read by the same application without having to open lots of different event queues.
- The PSPROP(NONE) configuration on the DEFINE SUB commands is to ensure that none of the message properties added by the publish/subscribe engine, for example MQTopicString, are added to the event message, ensuring that existing applications can continue to work unchanged.



Notes

# Viewing Event Messages

- Management or monitoring tool
  - Such as Tivoli Omegamon
- No general event formatter provided with WMQ
  - Several SupportPacs available
    - MO01 (C) - including source code
    - MS0K (C) - including source code
    - MS12 (COBOL) - including source code
    - MS0P – WMQ Explorer plug-in
    - MO71 – GUI Administrator



# Viewing Event Messages

- In order to view MQ Event messages we would normally expect there to be some management or monitoring tool to be used. However, event messages are just an MQ Message with a specific published format (which we will take a look at in a moment) so writing an application to view them is also possible.
- Apart from buying purpose built MQ monitoring tools, there are also a number of freely available MQ Support Pacs that allow the browsing and formatting of MQ event messages.
  - MO01 = Events and Dead Letter Queue Monitor
  - MS0K = Formatted dump of event queues in C – **COULD NOT FIND THIS ONE !!**
  - MS12 = WebSphere MQ for z/OS - Print Event Messages
  - MS0P = WebSphere MQ Explorer - Configuration and Display Extension Plug-ins
  - MO71 = GUI Administrator tool now marketed by MQGEM.
- MS0P and MO71 provide graphical tools that will format event messages (among various other tasks that they provide).
- Whether you buy or download a tool, or write your own application, something should be consuming the event messages that you configure you queue manager to emit, and taking appropriate action based on them.

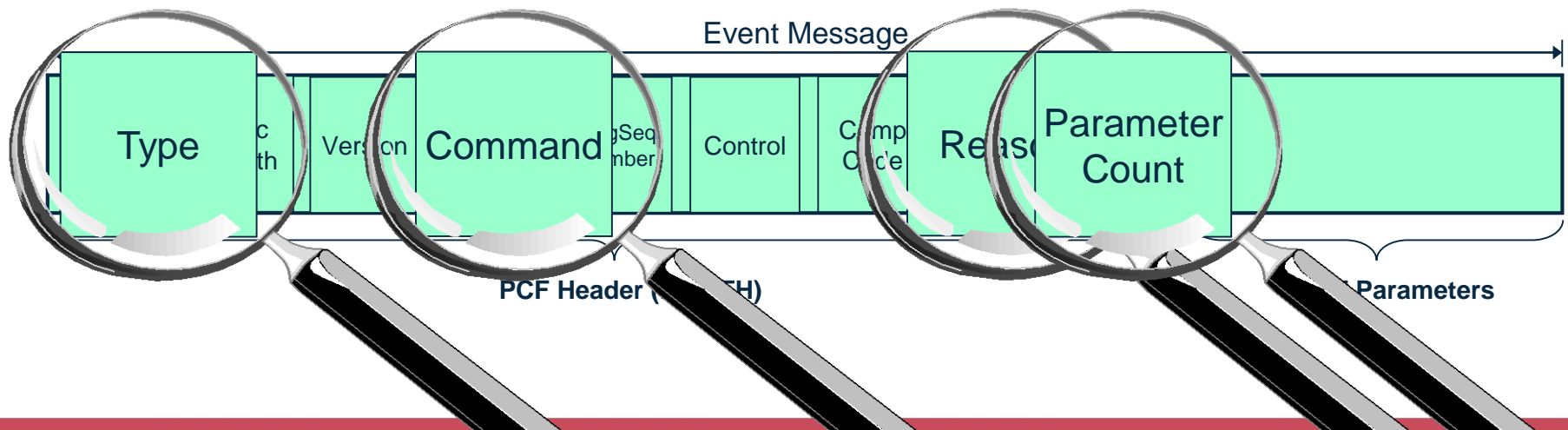


Notes



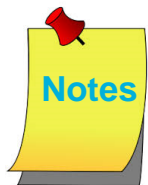
# Event Message – PCF Header

- PCF Header (MQCFH)
  - Type
    - MQCFT\_EVENT
  - StructLength, Version
    - Standard fields in all MQ headers
  - Command
    - MQCMD\_\*
    - Indicates what the event is
    - A value for each category of event
  - MsgSeqNumber, Control
    - Helpful with sets of related events
  - CompCode, Reason
    - MQCC\_WARNING
    - MQRC\_\*
    - A value for each type of event within the category
  - ParameterCount
    - How many pieces of data will follow



# Event Message – PCF Header - Notes

- We will look at the MQCFH header here and some example (common) sub-structures, and then for each specific event type later we will see more detail on the sort of data that follows in the sub-structures.
- The Type field tells you that this is an event message (MQCFT\_EVENT) since the PCF Header is actually used for other messages as well. PCF is a format that allows self-describing messages so it is very useful for lots of things.
- The StrucLength and Version fields are common to all MQ defined headers, allowing future-proofing against changes (extensions) to the structure.
- Our first important field is the Command field. This will contain a value that will indicate what category of event it is, with a defined MQCMD\_\* constant to use. This means that even if you have all your event messages aliased to a single central queue, you can still tell what they are. We will see the value used for the specific types of events when we come onto them. This is further qualified by Reason which we will see in a moment.



# Event Message – PCF Header - Notes

- MsgSeqNumber and Control are helpful when a single event is written as multiple messages. MsgSeqNumber contains an incrementing number of the event message within the set, and Control flags the last messages so you know when you have all the messages in the set. When a single event maps to a single message which is what happens most of the time, you will see MsgSeqNumber = 1, and Control = MQCFC\_LAST. We will see this used with one of the specific event types later.
- CompCode and Reason indicate the type of event within the category. CompCode is always MQCC\_WARNING, so Reason is the important one.
- ParameterCount tells you how much data to expect to see following this PCF header. The count will vary with event types and sometimes even within event types.



Notes

# PCF Header – C and COBOL

```
struct tagMQCFH {
    MQLONG  Type;           /* Structure type */
    MQLONG  StrucLength;   /* Structure length */
    MQLONG  Version;       /* Structure version number */
    MQLONG  Command;       /* Command identifier */
    MQLONG  MsgSeqNumber;  /* Message sequence number */
    MQLONG  Control;       /* Control options */
    MQLONG  CompCode;      /* Completion code */
    MQLONG  Reason;        /* Reason code qualifying completion code */
    MQLONG  ParameterCount; /* Count of parameter structures */
};
```

```
** MQCFH structure
10 MQCFH.
** Structure type
15 MQCFH-TYPE PIC S9(9) BINARY.
** Structure length
15 MQCFH-STRUCLength PIC S9(9) BINARY.
** Structure version number
15 MQCFH-VERSION PIC S9(9) BINARY.
** Command identifier
15 MQCFH-COMMAND PIC S9(9) BINARY.
** Message sequence number
15 MQCFH-MSGSEQNUMBER PIC S9(9) BINARY.
** Control options
15 MQCFH-CONTROL PIC S9(9) BINARY.
** Completion code
15 MQCFH-COMPCODE PIC S9(9) BINARY.
** Reason code qualifying completion code
15 MQCFH-REASON PIC S9(9) BINARY.
** Count of parameter structures
15 MQCFH-PARAMETERCOUNT PIC S9(9) BINARY.
```

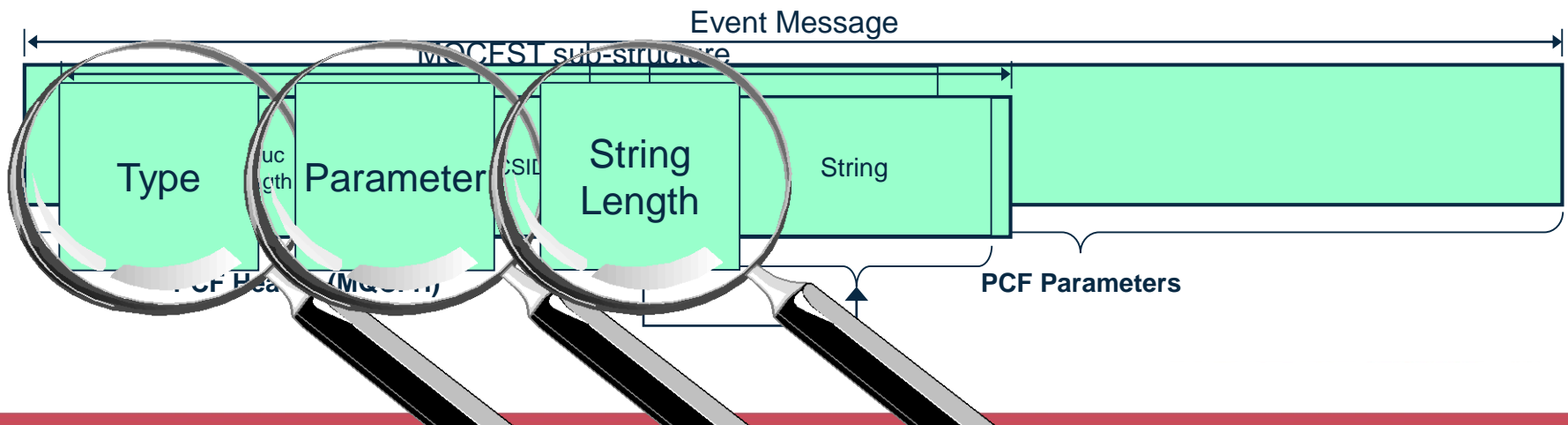


Notes

Complete your session evaluations online at [www.SHARE.org/Orlando-Eval](http://www.SHARE.org/Orlando-Eval)

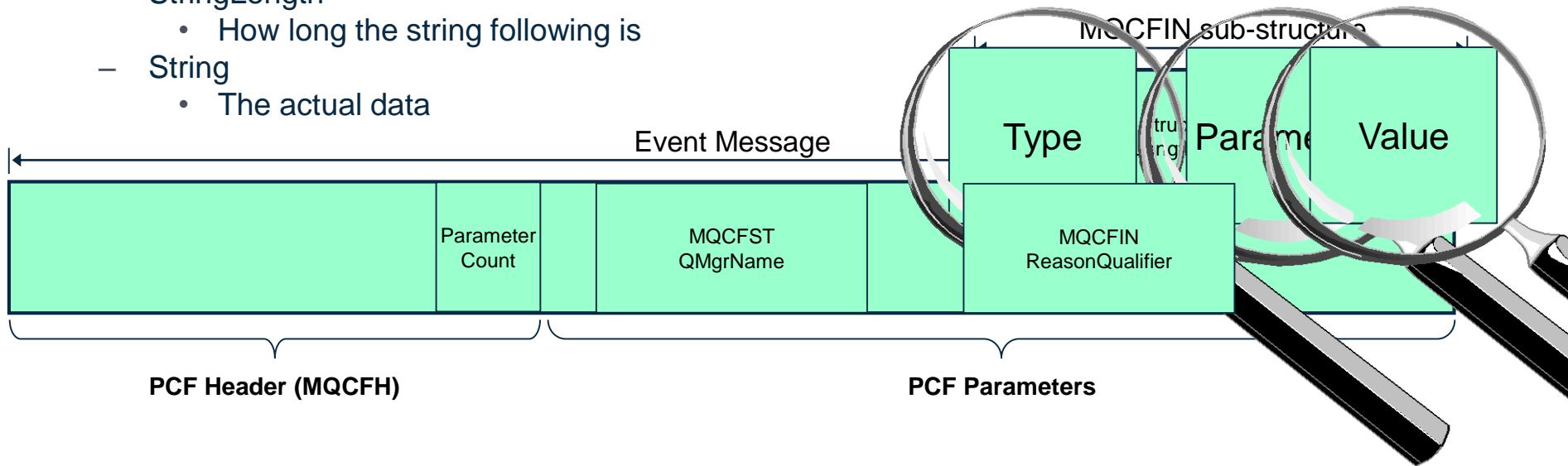
# Example parameters

- String (MQCFST) field example
  - Type
    - MQCFT\_STRING
  - StrucLength
  - Parameter
    - MQCA\_Q\_MGR\_NAME for our example
  - CodedCharSetId (CCSID)
    - The codepage that the string characters are represented in
  - StringLength
    - How long the string following is
  - String
    - The actual data



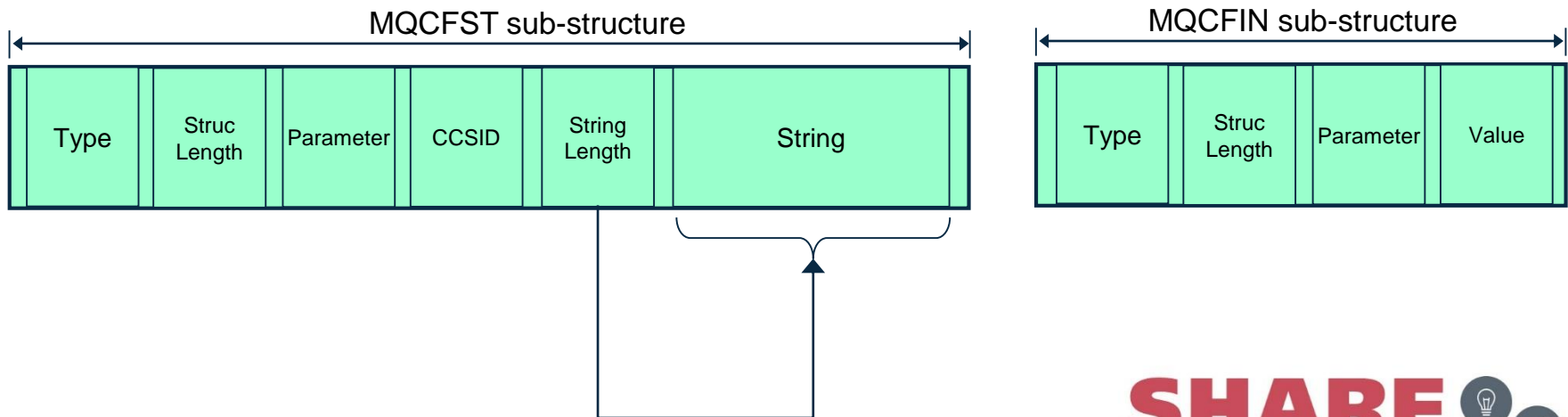
# Example parameters

- String (MQCFST) field example
  - Type
    - MQCFT\_STRING
  - StructLength
  - Parameter
    - MQCA\_Q\_MGR\_NAME for our example
  - CodedCharSetId (CCSID)
    - The codepage that the string characters are represented in
  - StringLength
    - How long the string following is
  - String
    - The actual data
- Integer field example
  - Type
    - MQCFT\_INTEGER
  - StructLength
  - Parameter
    - MQIACF\_REASON\_QUALIFIER
  - Value
    - The actual data
    - MQRQ\_\*



# Example parameters

- String (MQCFST) field example
  - Type
    - MQCFT\_STRING
  - StrucLength
  - Parameter
    - MQCA\_Q\_MGR\_NAME for our example
  - CodedCharSetId (CCSID)
    - The codepage that the string characters are represented in
  - StringLength
    - How long the string following is
  - String
    - The actual data
- Integer field example
  - Type
    - MQCFT\_INTEGER
  - StrucLength
  - Parameter
    - MQIACF\_REASON\_QUALIFIER
  - Value
    - The actual data
    - MQRQ\_\*



# Example Parameters

- We show a couple of example parameters here to give you an idea of the structure of the data in an event message. We have chosen one of each of the most common data types, a string parameter and an integer parameter.
- The example, the string QMgrName field is used by many different types of event messages as it records the name of the queue manager generating the event.
- Our second example, the integer ReasonQualifier field is used by a number of event types to further qualify the type of event within a category – we will see one of the events we look at later using it.



Notes

Complete your session evaluations online at [www.SHARE.org/Orlando-Eval](http://www.SHARE.org/Orlando-Eval)

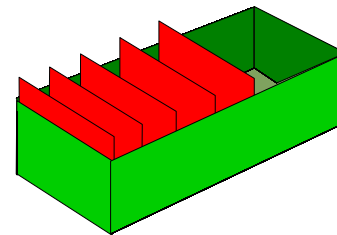


# Event Messages for Auditing

Complete your session evaluations online at [www.SHARE.org/Orlando-Eval](http://www.SHARE.org/Orlando-Eval)

# Security Failures

- Audit Trail of security access failures on your queue manager
- Queue Manager Attribute AUTHOREV
- PCF Header fields
  - Command
    - MQCMD\_Q\_MGR\_EVENT
  - Reasons
    - MQRC\_NOT\_AUTHORIZED
  - Reason Qualifier
    - MQRQ\_CONN\_NOT\_AUTHORIZED
    - MQRQ\_OPEN\_NOT\_AUTHORIZED
    - MQRQ\_CLOSE\_NOT\_AUTHORIZED
    - MQRQ\_CMD\_NOT\_AUTHORIZED
    - MQRQ\_SUB\_NOT\_AUTHORIZED
    - MQRQ\_SUB\_DEST\_NOT\_AUTHORIZED



**SYSTEM.ADMIN.QMGR.EVENT**

**ALTER QMGR AUTHOREV(ENABLED)**

# Security Failures – Notes

- On the distributed platforms, an audit trail of access failures is kept by means of event messages which are written to the SYSTEM.ADMIN.QMGR.EVENT queue. You can enable these events to be written by means of the AUTHOREV switch on ALTER QMGR.
- There are several different types of MQRQ\_NOT\_AUTHORIZED events showing specifically what kind of access was attempted. Each of these types has a different reason qualifier recorded in the event message.
  - MQRQ\_CONN\_NOT\_AUTHORIZED
  - MQRQ\_OPEN\_NOT\_AUTHORIZED
    - MQPUT1 ==> MQOPEN
  - MQRQ\_CLOSE\_NOT\_AUTHORIZED
    - For deletion of dynamic queues
  - MQRQ\_CMD\_NOT\_AUTHORIZED
    - IBM MQ MQSC/PCF commands
  - MQRQ\_SUB\_NOT\_AUTHORIZED
    - subscribe check failed
  - MQRQ\_SUB\_DEST\_NOT\_AUTHORIZED
    - destination queue check failed
- and, where applicable, there is information in each event message to show the user ID and application that made the failed access attempt.



Notes

Complete your session evaluations online at [www.SHARE.org/Orlando-Eval](http://www.SHARE.org/Orlando-Eval)

# Not Authorized Event Message Details

Reason Qualifier	QMgr Name	User Identifier	Appl Type	Appl Name	Options	Dest Open Options	Object QMgr Name	Object Name *	Topic String	Admin Topic Names	Command
MQRQ_CONN_NOT_AUTHORIZED	X	X	X	X							
MQRQ_OPEN_NOT_AUTHORIZED	X	X	X	X	X		X	X	X	X	
MQRQ_CLOSE_NOT_AUTHORIZED	X	X	X	X				X	X	X	
MQRQ_CMD_NOT_AUTHORIZED	X	X									X
MQRQ_SUB_NOT_AUTHORIZED	X	X	X	X	X				X	X	
MQRQ_SUB_DEST_NOT_AUTHORIZED	X	X	X	X	X	X	X	X			

\* Might be Q Name, Process Name (OPEN only), Namelist Name (OPEN only), Sub Name (CLOSE only)

# Not Authorized Event Message Details - Notes

- Much of the contents of the Not Authorized event messages are fairly self-explanatory, but we will look at a few of the interesting fields in a bit more detail. Of course you can read about the details of event messages in the “Monitoring WebSphere MQ” book too.
  - [http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/topic/com.ibm.mq.csqzax.doc/mo10120\\_.htm](http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/topic/com.ibm.mq.csqzax.doc/mo10120_.htm)
- The not authorised events that are from API calls, all except MQRQ\_CMD\_NOT\_AUTHORIZED, provide details of the application making the call as well as the user ID that the application was running under that did not have the required access. These application identity fields are the same fields you will see on DISPLAY CONN, APPLTAG and APPLTYPE along with USERID.
- MQOPEN and MQSUB authorisation failures will provide the options used on the verb to allow you to work out what access was attempted on the resource. These are integer fields that contain all the options and can be decoded most easily working in hex (or of course using a tool such as in MS0P or MO71 to decode these for you).



Notes

# Identifying the Application

- Event message contains same info you'd see on DISPLAY CONN command
  - APPLTAG
  - APPLTYPE
  - USERID (Not Authorized events only)

## AMQ8276: Display Connection details.

```

CONN(3C75FA4B2001BA01)
TYPE(CONN)
PID(9428)                TID(1)
APPLTAG(d:\nttools\q.exe) APPLTYPE(USER)
USERID(hughson)
CHANNEL( )              CONNAME( )
CONNOPTS(MQCNO_SHARED_BINDING)
  
```

User Identifier	Appl Type	Appl Name
X	X	X
X	X	X
X	X	X
X		
X	X	X
X	X	X

# Decoding Options

- An integer field containing the options used
- Examples

## Options (from MQOPEN) 0x00000012

MQOO\_OUTPUT                    0x00000010

MQOO\_INPUT\_SHARED            0x00000002

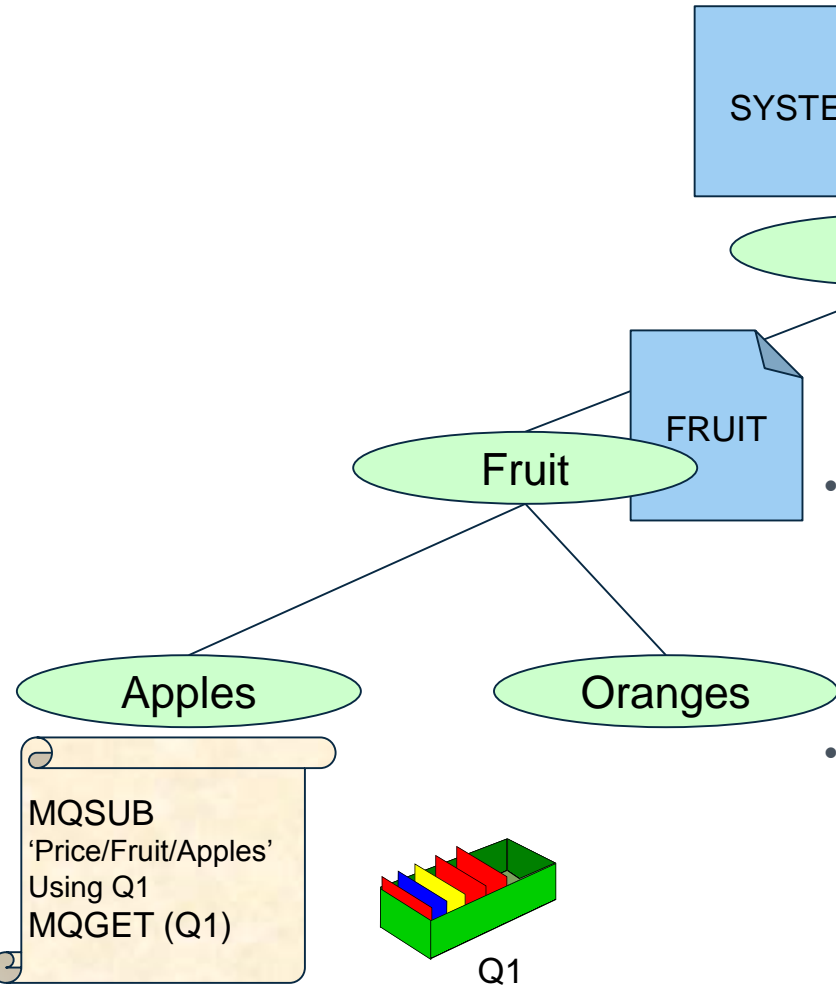
## Options (from MQSUB) 0x0000000A

MQSO\_CREATE                    0x00000002

MQSO\_DURABLE                   0x00000008

Options	Dest Open Options
X	
X	
X	X

# Topic Security



- Authority check on topic objects
  - “Walk up the tree”
  - May be more than one check
  - Failed event records all topic objects used for authority check
- Authority check on destination queue
  - When not using MQSO\_MANAGED
  - Check is for PUT to that queue

Topic String	Admin Topic Names
X	X
X	X
X	X



# Topic Security

- When MQOPENing a topic (MQOT\_TOPIC) for MQOO\_OUTPUT – that is, in order to publish, or when making an MQSUB call to subscribe to a topic, a security check is done to see if your user ID has authority to use that topic.
- In our example we have called MQSUB at the point in the topic tree, “Price/Fruit/Apples”. There is no topic object at this point in the topic tree, so to find the profile we need to check authorities against we walk up the topic tree to find a node which does have a topic object. The next point is “Price/Fruit”. This does have a topic object, FRUIT, so we will check that this user ID has subscribe authority on the profile for the FRUIT topic. If that user ID does have authority, our search stops there. If it does not, we carry on searching up the topic tree and will check the SYSTEM.BASE.TOPIC to see if this user ID has subscribe authority there.
- An additional authorisation check is done for an MQSUB call when the application wishes to use a specific destination queue (i.e. is not using the MQSO\_MANAGED option). In this case we also check that this user ID has authority to GET from that destination queue.

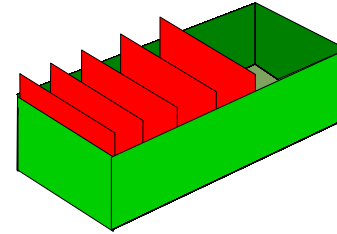


Notes

Complete your session evaluations online at [www.SHARE.org/Orlando-Eval](http://www.SHARE.org/Orlando-Eval)

# Commands Issued

- Audit Trail of MQSC/PCF commands issued on your queue manager
- Queue Manager Attribute CMDEV
  - NODISPLAY
- Command Failed => No event
- PCF Header fields
  - Command
    - MQCMD\_COMMAND\_EVENT
  - Possible Reasons
    - MQRC\_COMMAND\_MQSC
    - MQRC\_COMMAND\_PCF



**SYSTEM.ADMIN.COMMAND.EVENT**

**ALTER QMGR CMDEV(ENABLED)**

# Commands Issued - Notes

- An audit trail of commands issued is kept by means of event messages which are written to the SYSTEM.ADMIN.COMMAND.EVENT queue. You can enable these events to be written by means of the CMDEV switch on ALTER QMGR.
- You can choose to record all commands that are issued, or perhaps more usefully, all commands except DISPLAY commands (PCF Inquire commands), so that you only capture a record of those potentially destructive or interesting commands. This is done using CMDEV(NODISPLAY).
- If the command issued failed, for example a syntax error, then no command event is generated.
- The PCF Header of a command event message will record the Command field as MQCMD\_COMMAND\_EVENT and can have one of two possible Reasons, MQRC\_COMMAND\_MQSC or MQRC\_COMMAND\_PCF. The next page will show the differences.
- Command events are available on z/OS in V6 and Distributed platforms in V7.0.1.
- SupportPac MS0P also provides an API Exit that can be used to log all commands sent to the Command Server on the Distributed platforms for earlier versions.



Notes

Complete your session evaluations online at [www.SHARE.org/Orlando-Eval](http://www.SHARE.org/Orlando-Eval)

# Command Event Message Details

Event Origin	Command Context							Command	Command Data *
	Event User ID	Event QMgr	Event Accounting Token	Event Identity Data	Event Appl Type	Event Appl Name	Event Appl Origin		
<b>MQEVO_CONSOLE</b>	X	X						X	X
<b>MQEVO_INIT</b>	X	X						X	X
<b>MQEVO_MSG</b>	X	X	X	X	X	X	X	X	X
<b>MQEVO_INTERNAL</b>	X	X						X	X
<b>MQEVO_OTHER</b>	X	X						X	X
* Either PCF message as it was submitted, or text of MQSC command									

# Command Event Message Details - Notes

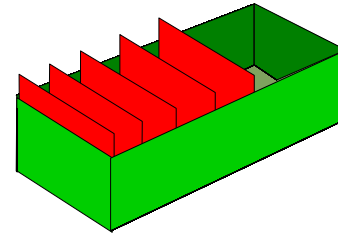
- Much of the contents of the Command event messages are fairly self-explanatory, but we will look at a few of the interesting fields in a bit more detail. Of course you can read about the details of event messages in the “Monitoring WebSphere MQ” book too.
  - [http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/topic/com.ibm.mq.cs.qzax.doc/mo10120\\_.htm](http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/topic/com.ibm.mq.cs.qzax.doc/mo10120_.htm)
- The contents of the command event message varies depending on how the command was issued. If the command was a PCF message then the content of the input PCF message is part of the command event. Alternatively, if the command was an MQSC message then this text string will be found in the event message instead of the PCF input message.
- If the command was issued by putting a message on the command server queue (MQEVO\_MSG) then there will be more application identifying information than in other cases because the Message Descriptor (MQMD) of the command message written by the application contains lots of extra data.
- In all cases you will get the user ID issuing the command, the queue manager where the command was entered, and one of the two aforementioned command data variants.



Notes

# Configuration Changes

- Audit trail of changes to the configuration of the queue manager.
  - Commands acting on objects
  - MQSET calls
- Queue Manager Attribute CONFIGEV
- Create a base-line view with REFRESH QMGR
- PCF Header fields
  - Command
    - MQCMD\_CONFIG\_EVENT
  - Possible Reasons
    - MQRC\_CONFIG\_CHANGE\_OBJECT
    - MQRC\_CONFIG\_CREATE\_OBJECT
    - MQRC\_CONFIG\_DELETE\_OBJECT
    - MQRC\_CONFIG\_REFRESH\_OBJECT



**SYSTEM.ADMIN.CONFIG.EVENT**

**ALTER QMGR CONFIGEV(ENABLED)**

**REFRESH QMGR TYPE(CONFIGEV)  
OBJECT(ALL) NAME(\*)**

# Configuration Changes - Notes

- An audit trail of changes to the queue manager configuration is kept by means of event messages which are written to the SYSTEM.ADMIN.CONFIG.EVENT queue. You can enable these events to be written by means of the CONFIGEV switch on ALTER QMGR.
- These events will be generated when a DEFINE, ALTER or DELETE command acts upon an object, or an MQSET command is used.
- A base-line picture of the current queue manager configuration can be created by using the REFRESH QMGR TYPE(CONFIGEV) command which will create an event message for every object in the queue manager. Since this could be a heavyweight operation if you have a lot of objects, you can break it down into smaller sets of objects using the NAME and OBJECT qualifiers on the command.
- The PCF Header of a configuration event message will record the Command field as MQCMD\_CONFIG\_EVENT and can have one of four possible Reasons, MQRC\_CONFIG\_CHANGE\_OBJECT, MQRC\_CONFIG\_CREATE\_OBJECT or MQRC\_CONFIG\_DELETE\_OBJECT for the respective MQSC or PCF commands that you might issue upon an object or MQRC\_CONFIG\_REFRESH\_OBJECT for those event messages written when creating the base-line picture. The next page will show the details.
- Config events are available on z/OS in V5.3 and Distributed platforms in V7.0.1.



Notes

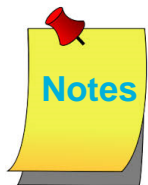
# Config Event Message Details

- All config events
  - Command Context just as [Command Event Message Details](#)
  - Object Type
  - Object Name
  - Disposition (z/OS only)
- PCF Header
  - When 2 event messages
  - MsgSeqNumber = 1, 2
  - Control = MQCFC\_NOT\_LAST, MQCFC\_LAST
- MQRC\_CONFIG\_CHANGE\_OBJECT
  - 2 event messages
  - Attributes before change
  - Attributes after change
- MQRC\_CONFIG\_CREATE\_OBJECT
  - 1 event message
  - Attributes after create
- MQRC\_CONFIG\_DELETE\_OBJECT
  - 1 event message
  - Attributes before deletion
- MQRC\_CONFIG\_REFRESH\_OBJECT
  - 1 event message
  - Current attributes of object



# Config Event Message Details - Notes

- Much of the contents of the Config event messages contain object attributes. These are PCF sub-structures when one issues a PCF command to make or alter an object, and exactly the same sub-structures are used in these event messages. Of course you can read about the details of event messages in the “Monitoring WebSphere MQ” book too.
  - [http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/topic/com.ibm.mq.csqzax.doc/mo10120 .htm](http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/topic/com.ibm.mq.csqzax.doc/mo10120.htm)
- The contents of the command event message varies depending on how the command was issued just as with command events. If the command was issued by putting a message on the command server queue (MQEVO\_MSG) then there will be more application identifying information than in other cases because the Message Descriptor (MQMD) of the command message written by the application contains lots of extra data.
- In all cases you will get the user ID issuing the command, the queue manager where the command was entered.
- In the specific case of the MQRC\_CONFIG\_CHANGE\_OBJECT, you will get two messages, one containing the object attributes before the change and one containing those after the change. These will be indicated using the MsgSeqNumber field and the Control field in the PCF Header.

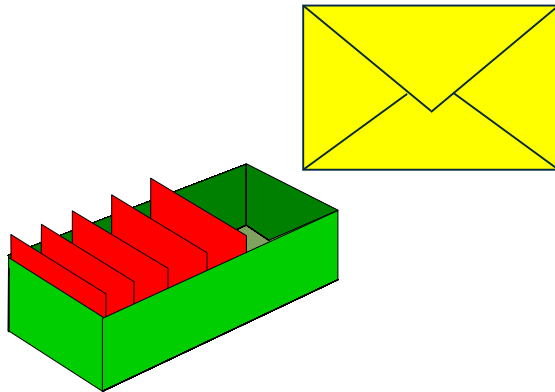


# Combining Command and Config Events

- ALTER QLOCAL(FRED) MAXDEPTH(1)
  - Command Event
  - Before Change Config Event
  - After Change Config Event

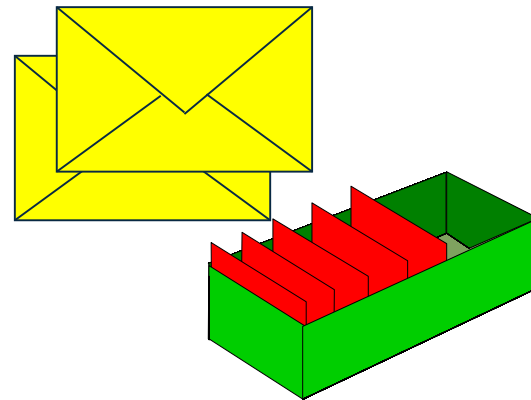
```
ALTER QMGR  
CMDEV(NODISPLAY)  
CONFIGEV(ENABLED)
```

Correl ID = 1234



SYSTEM.ADMIN.COMMAND.EVENT

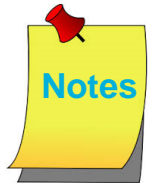
Correl ID = 1234



SYSTEM.ADMIN.CONFIG.EVENT

# Combining Command and Config Events - Notes

- If you have both Command events and Configuration events enabled, then when an object is changed, the event messages will share the same correlation ID in their MQMDs.



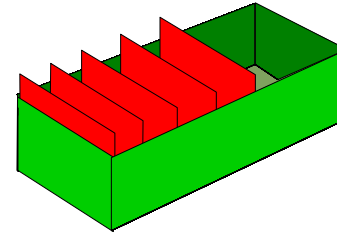
Complete your session evaluations online at [www.SHARE.org/Orlando-Eval](http://www.SHARE.org/Orlando-Eval)

# Event Messages for Monitoring

Complete your session evaluations online at [www.SHARE.org/Orlando-Eval](http://www.SHARE.org/Orlando-Eval)

# Start and Stop Events

- Notification of queue manager start and stop
- Queue Manager Attribute STRSTPEV
  - DEFPSIST of event queue
  
- PCF Header fields
  - Command
    - MQCMD\_Q\_MGR\_EVENT
  
  - Possible Reasons
    - MQRC\_Q\_MGR\_ACTIVE
    - MQRC\_Q\_MGR\_NOT\_ACTIVE
  
  - Reason Qualifier
    - MQRQ\_Q\_MGR\_STOPPING
    - MQRQ\_Q\_MGR\_QUIESCING



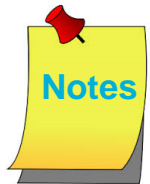
**SYSTEM.ADMIN.QMGR.EVENT**

**ALTER QMGR STRSTPEV(ENABLED)**

Reason	QMGR Name	Reason Qualifier
MQRC_Q_MGR_ACTIVE	X	
MQRC_Q_MGR_NOT_ACTIVE	X	X

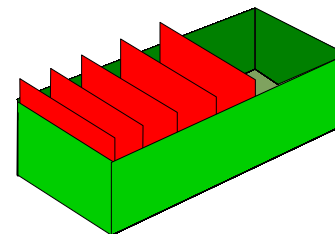
# Start and Stop Events - Notes

- Whenever a queue manager starts or stops, an event message is written to the SYSTEM.ADMIN.QMGR.EVENT queue. You can enable these events to be written by means of the STRSTPEV switch on ALTER QMGR.
- The PCF Header of a start or stop event message will record the Command field as MQCMD\_Q\_MGR\_EVENT and can have two possible Reasons where the MQRC\_Q\_MGR\_NOT\_ACTIVE can have two possible Reason Qualifiers as shown.
- A stop event is only recorded if the DEFPSIST attribute of the SYSTEM.ADMIN.QMGR.EVENT queue is defined as persistent.



# Channel Events

- Notification of channel starts, stops and errors, including SSL errors
- Queue Manager Attributes CHLEV and SSLEV
  - EXCEPTION



**SYSTEM.ADMIN.CHANNEL.EVENT**

**ALTER QMGR CHLEV(EXCEPTION)  
SSLEV(ENABLED)**

- Server-connection channels do not cause start or stop events
- PCF Header fields
  - Command
    - MQCMD\_CHANNEL\_EVENT
  - Possible Reasons
    - MQRC\_CHANNEL\_ACTIVATED
    - MQRC\_CHANNEL\_NOT\_ACTIVATED
    - MQRC\_CHANNEL\_STARTED
    - MQRC\_CHANNEL\_STOPPED
    - MQRC\_CHANNEL\_STOPPED\_BY\_USER
    - MQRC\_CHANNEL\_CONVERSION\_ERROR
    - MQRC\_CHANNEL\_SSL\_ERROR
    - MQRC\_CHANNEL\_SSL\_WARNING

- Reason Qualifier
  - MQRQ\_CHANNEL\_STOPPED\_OK
  - MQRQ\_CHANNEL\_STOPPED\_ERROR
  - MQRQ\_CHANNEL\_STOPPED\_RETRY
  - MQRQ\_CHANNEL\_STOPPED\_DISABLED
  - MQRQ\_SSL\_HANDSHAKE\_ERROR
  - MQRQ\_SSL\_CIPHER\_SPEC\_ERROR
  - MQRQ\_SSL\_PEER\_NAME\_ERROR
  - MQRQ\_SSL\_CLIENT\_AUTH\_ERROR
  - MQRQ\_SSL\_UNKNOWN\_REVOCATION

# Channel Events - Notes

- Whenever a channel does something noteworthy, an event message is written to the `SYSTEM.ADMIN.CHANNEL.EVENT` queue. You can enable these events to be written by means of the `CHLEV` switch on `ALTER QMGR`. Additionally, if you are using SSL on your channels, you can get detailed event messages on the reason for any SSL failures by enabling SSL error events by means of the `SSLEV` switch on `ALTER QMGR`. These are written to the same event queue.
- You can choose to record all channel events that are issued, or perhaps more usefully, all the ones relating to error situations. This is done using `CHLEV(EXCEPTION)`.
- The PCF Header of a channel event message will record the Command field as `MQCMD_CHANNEL_EVENT` and can have several possible Reasons and Reason Qualifiers as shown.

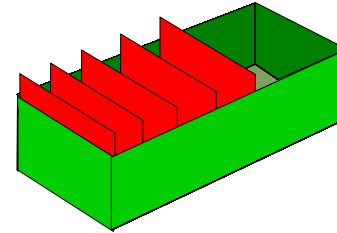


Notes



# Channel Auto-definition Events

- Notification of attempts to automatically define channels
- Queue Manager Attributes
  - CHAD
  - CHADEV
  - CHADEXIT
- PCF Header fields
  - Command
    - MQCMD\_CHANNEL\_EVENT
  - Possible Reasons
    - MQRC\_CHANNEL\_AUTO\_DEF\_ERROR
    - MQRC\_CHANNEL\_AUTO\_DEF\_OK

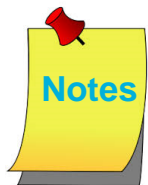


**SYSTEM.ADMIN.CHANNEL.EVENT**

```
ALTER QMGR CHAD(ENABLED)  
CHADEV(ENABLED)  
CHADEXIT(exit-name)
```

# Channel Auto-definition Events - Notes

- If you have enabled channel auto-definition by means of the CHAD switch on ALTER QMGR, then an inbound channel connection who wishes to use a channel that is not defined will cause it to be created (modeled off SYSTEM.AUTO.\* channels). You can further configure this behaviour with an exit whose name is specified in the CHADEXIT attribute, and you can also have events emitted whenever an auto-definition of a channel is attempted.
- Whenever a channel is automatically defined, an event message is written to the SYSTEM.ADMIN.CHANNEL.EVENT queue. You can enable these events to be written by means of the CHADEV switch on ALTER QMGR. The PCF Header of a channel event message will record the Command field as MQCMD\_CHANNEL\_EVENT and can have one of two possible Reasons, MQRC\_CHANNEL\_AUTO\_DEF\_OK or MQRC\_CHANNEL\_AUTO\_DEF\_ERROR.



# Channel Event Message Details

Reason	QMgr Name	Channel Name	XmitQ Name	Connection Name	Error Identifier	Aux Error Data Int 1/2	Aux Error Data String 1/2/3	Conversion Reason Code	Format	Channel Type	SSL Handshake Stage	SSL Return Code	SSL Peer Name
MQRC_CHANNEL_ACTIVATED	X	X	X	X									
MQRC_CHANNEL_NOT_ACTIVATED	X	X	X	X									
MQRC_CHANNEL_STARTED	X	X	X	X									
MQRC_CHANNEL_STOPPED	X	X	X	X	X	X	X						
MQRC_CHANNEL_STOPPED_BY_USER	X	X	X	X	X								
MQRC_CHANNEL_CONV_ERROR	X	X	X	X				X	X				
MQRC_CHANNEL_SSL_ERROR	X	X	X	X							X	X	X
MQRC_CHANNEL_SSL_WARNING	X	X	X	X									
MQRC_CHANNEL_AUTO_DEF_OK	X	X		X						X			
MQRC_CHANNEL_AUTO_DEF_ERROR	X	X		X	X	X				X			

# Identifying the Channel

- Event message contains same info as on the DISPLAY CHSTATUS command
  - Channel name
  - XMITQ
  - CONNAME

```

AMQ8417: Display Channel Status details.
CHANNEL(NT1MAH.TO.NT2MAH)      CHLTYPE(SDR)
CONNAME(127.0.0.1(1502))       CURRENT
RQMNAME(NT2MAH)                STATUS(RUNNING)
SUBSTATE(MQGET)                 XMITQ(NT2MAH)
    
```

Channel Name	Xmitq Name	Connection Name
X	X	X
X	X	X
X	X	X
X	X	X
X	X	X
X	X	X
X	X	X
X	X	X
X		X
X		X

# Decoding Error Information

- Error Identifier
  - Example  
`0x20009208`
- Use mqrc tool
  - Can enter number in hex or decimal
- Auxiliary data
  - Contains data that would be seen in the inserts of the equivalent error log message

**Receive Failed**

```

C:\>mqrc 0x20009208

536908296 0x20009208 rrcE_RECEIVE_FAILED

C:\>mqrc AMQ9208

MESSAGE:
Error on receive from host <insert one>.

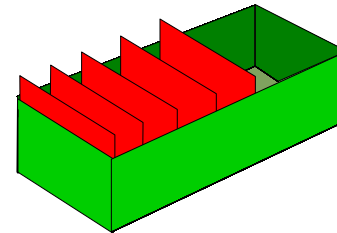
EXPLANATION:
An error occurred receiving data from <insert one>
over <insert two>. This may be due to a
communications failure.

ACTION:
The return code from the <insert two><insert three>
call was 1111 (X'8AE'). Record these values and tell
the systems administrator.
  
```

Error Identifier	Aux Error Data Int 1/2	Aux Error Data String 1/2/3
X	X	X
X		
X	X	

# Events for errors returned to applications

- Notification of application failures to use local or remote queues, inhibited resources
- Queue Manager Attributes
  - LOCALEV
  - REMOTEEV
  - INHIBITEV
- PCF Header fields
  - Command
    - MQCMD\_Q\_MGR\_EVENT
  - Possible Reasons
    - MQRC\_ALIAS\_BASE\_Q\_TYPE\_ERROR
    - MQRC\_UNKNOWN\_ALIAS\_BASE\_Q
    - MQRC\_UNKNOWN\_OBJECT\_NAME
    - MQRC\_DEF\_XMIT\_Q\_TYPE/USAGE\_ERROR
    - MQRC\_Q\_TYPE\_ERROR
    - MQRC\_REMOTE\_Q\_NAME\_ERROR
    - MQRC\_XMIT\_Q\_TYPE/USAGE\_ERROR
    - MQRC\_UNKNOWN\_DEF\_XMIT\_Q
    - MQRC\_UNKNOWN\_REMOTE\_Q\_MGR
    - MQRC\_UNKNOWN\_XMIT\_Q
    - MQRC\_GET\_INHIBITED
    - MQRC\_PUT\_INHIBITED
- Reflects an MQRC also given back to the application
- Handy when application forgets to check!



**SYSTEM.ADMIN.QMGR.EVENT**

**ALTER QMGR LOCALEV(ENABLED)  
REMOTEEV(ENABLED)  
INHIBITEV(ENABLED)**

# Events for errors returned to applications

- These events reflect errors that are also returned to applications. They indicate problems where either the object that the application needs to use is defined incorrectly, or the application is using the wrong name of an object, perhaps more likely in cases such as MQRC\_UNKNOWN\_OBJECT\_NAME.
- The inhibit events reflect a problem where an application has attempted to use a resource which is inhibited for the operation required. MQRC\_PUT\_INHIBIT events are created for PUT(DISABLED) queues and PUB(DISABLED) topics.
- These events can also be generated by internal operations in the queue manager, such as the writing of an event message. In case it would reflect the ReplyToQ and ReplyToQMgr fields of an originating application message containing incorrect information or pointing at a resource that is not correctly defined.



Notes

Complete your session evaluations online at [www.SHARE.org/Orlando-Eval](http://www.SHARE.org/Orlando-Eval)

# Local/Remote/Inhibit Event Message Details



Reason	QMgr Name	Object Name *	Object QMgr Name	Base Object Name	Xmit Q Name	Base Type	Q Type	Appl Type	Appl Name	Topic String
MQRC_ALIAS_BASE_Q_TYPE_ERROR	X	X	X	X			X	X	X	
MQRC_UNKNOWN_ALIAS_BASE_Q	X	X	X	X		X		X	X	
MQRC_UNKNOWN_OBJECT_NAME	X	X	X					X	X	
MQRC_DEF_XMIT_Q_TYPE_ERROR	X	X	X		X		X	X	X	
MQRC_DEF_XMIT_Q_USAGE_ERROR	X	X	X		X			X	X	
MQRC_Q_TYPE_ERROR	X	X	X					X	X	
MQRC_REMOTE_Q_NAME_ERROR	X	X	X					X	X	
MQRC_XMIT_Q_TYPE_ERROR	X	X	X		X		X	X	X	
MQRC_XMIT_Q_USAGE_ERROR	X	X	X		X			X	X	
MQRC_UNKNOWN_DEF_XMIT_Q	X	X	X		X			X	X	
MQRC_UNKNOWN_REMOTE_Q_MGR	X	X	X					X	X	
MQRC_UNKNOWN_XMIT_Q	X	X	X		X			X	X	
MQRC_GET_INHIBITED	X	X						X	X	
MQRC_PUT_INHIBITED	X	X	X					X	X	X

\* Might be Q Name, Process Name, Namelist Name or Topic Name

Complete your session evaluations online at [www.SHARE.org/Orlando-Eval](http://www.SHARE.org/Orlando-Eval)





# Local/Remote/Inhibit Event Message Details - Notes

- Much of the contents of these queue manager event messages are fairly self-explanatory, but we will look at a few of the interesting fields in a bit more detail. Of course you can read about the details of event messages in the “Monitoring WebSphere MQ” book too.
  - [http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/topic/com.ibm.mq.cs.qzax.doc/mo10120 .htm](http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/topic/com.ibm.mq.cs.qzax.doc/mo10120.htm)
- These events are all caused by some application making an MQ API call (mostly MQOPENS) which failed. So they all provide details of the application making the call. These application identity fields are the same fields you will see on DISPLAY CONN, APPLTAG and APPLTYPE. We saw an example of this earlier with the Not Authorized events.
- When diagnosing an issue reported by one of these events, either the application has coded the wrong thing, or there is an administrative definition missing or incorrect that the application needs to use. You can see the details of exactly what the application coded in the MQOD.ObjectName and MQOD.ObjectQMgrName fields by looking at the equivalent fields in the event message.



Notes

# Application MQOD details

- MQOPEN API call
- MQOD describes the object to open
  - ObjectName
  - ObjectQMgrName
- MQMD describes the object to open for a Response/Report message
  - ReplyToQ
  - ReplyToQMgr
- Event message data provides these

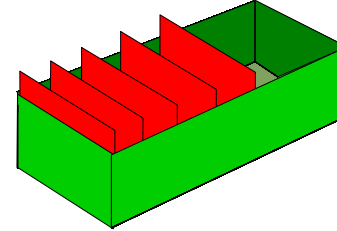
```
MQOD   ObjDesc = {MQOD_DEFAULT};

ObjDesc.ObjectType           = MQOT_Q;
strncpy (ObjDesc.ObjectName,
         "MY.APP.QUEUE" ,
         MQ_Q_NAME_LENGTH) ;
```

Object Name *	Object QMgr Name
X	X
X	X
X	X
X	X
X	X
X	X
X	X

# Queue Service Interval Events

- Notification of whether queues are being processed in a timely manner
  - MQGETs or CLEAR QLOCAL
  
- Queue Manager Attribute PERFMEV
- Queue Attributes
  - QSVCI EV – Event switch
  - QSVCI NT – Interval
  
- PCF Header fields
  - Command
    - MQCMD\_PERFM\_EVENT
  
  - Possible Reasons
    - MQRC\_Q\_SERVICE\_INTERVAL\_HIGH
    - MQRC\_Q\_SERVICE\_INTERVAL\_OK



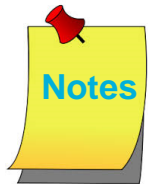
**SYSTEM.ADMIN.PERFM.EVENT**

**ALTER QMGR PERFMEV(ENABLED)**

**ALTER QLOCAL(*q-name*)  
QSVCI EV(HIGH)  
QSVCI NT(10000)**

# Queue Service Interval Events – Notes

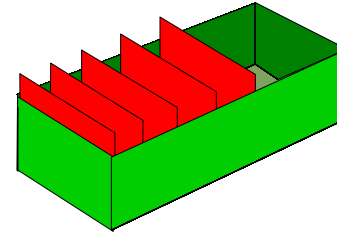
- Queue Service Interval events give you notification that a queue is not being processed in a timely enough manner. You can set the interval on the queue (in milliseconds) for how quickly you expect gets to be issued after some messages are there to be processed, and if a get hasn't been used within that interval you will get an event.
- A CLEAR QLOCAL command will also count as a 'get' for the purposes of these events.
- Both these High/Low events and Queue Depth High/Low events work in co-operation with each other as we will see in a moment.



Complete your session evaluations online at [www.SHARE.org/Orlando-Eval](http://www.SHARE.org/Orlando-Eval)

# Queue Depth Events

- Notification of when queues start to fill up with messages
- Queue Manager Attribute PERFMEV
- Queue Attributes
  - QDPHIEV – High Depth Event Switch
  - QDPLOEV – Low Depth Event Switch
  - QDPMAXEV – Max Depth Event Switch
  
  - QDEPTHHI - % of Max Depth
  - QDEPTHLO - % of Max Depth
  - MAXDEPTH
- PCF Header fields
  - Command
    - MQCMD\_PERFM\_EVENT
  
  - Possible Reasons
    - MQRC\_Q\_DEPTH\_HIGH
    - MQRC\_Q\_DEPTH\_LOW
    - MQRC\_Q\_FULL



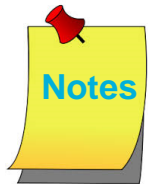
**SYSTEM.ADMIN.PERFM.EVENT**

```
ALTER QMGR PERFMEV(ENABLED)
```

```
ALTER QLOCAL(q-name)  
MAXDEPTH(1000)  
QDEPTHHI(80)  
QDEPTHLO(20)  
QDPHIEV(ENABLED)
```

# Queue Depth Events – Notes

- Queue Depth events give you prior warning that your queue is getting full. A queue full event of course tells you that the queue is full – too late!! However, you can get prior warning by using the Queue High event, and setting the point you wish to be notified as a percentage of the maximum depth of the queue.
- There are also Queue Low events which tell you when the panic is over and the alarms can be turned off again!
- Both these High/Low events and Queue Service Interval High/Low events work in co-operation with each other as we will see in a moment.



Complete your session evaluations online at [www.SHARE.org/Orlando-Eval](http://www.SHARE.org/Orlando-Eval)

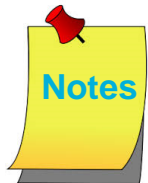
# Performance Event Message Details

Reason	QMgr Name	Q Name	Statistics			
			Time Since Reset	High Q Depth	Msg Enq Count	Msg Deq Count
MQRC_Q_SERVICE_INTERVAL_HIGH	X	X	X	X	X	X
MQRC_Q_SERVICE_INTERVAL_OK	X	X	X	X	X	X
MQRC_Q_DEPTH_HIGH	X	X	X	X	X	X
MQRC_Q_DEPTH_LOW	X	X	X	X	X	X
MQRC_Q_FULL	X	X	X	X	X	X

- Performance Event Statistics are reset when:
  - A performance event is emitted
  - The queue manager restarts
  - Reset Queue Statistics (PCF) command is issued
  - RESET QSTATS (MQSC) command is issued (z/OS only)
- Time Since Reset
  - Shows time of last one of the above actions

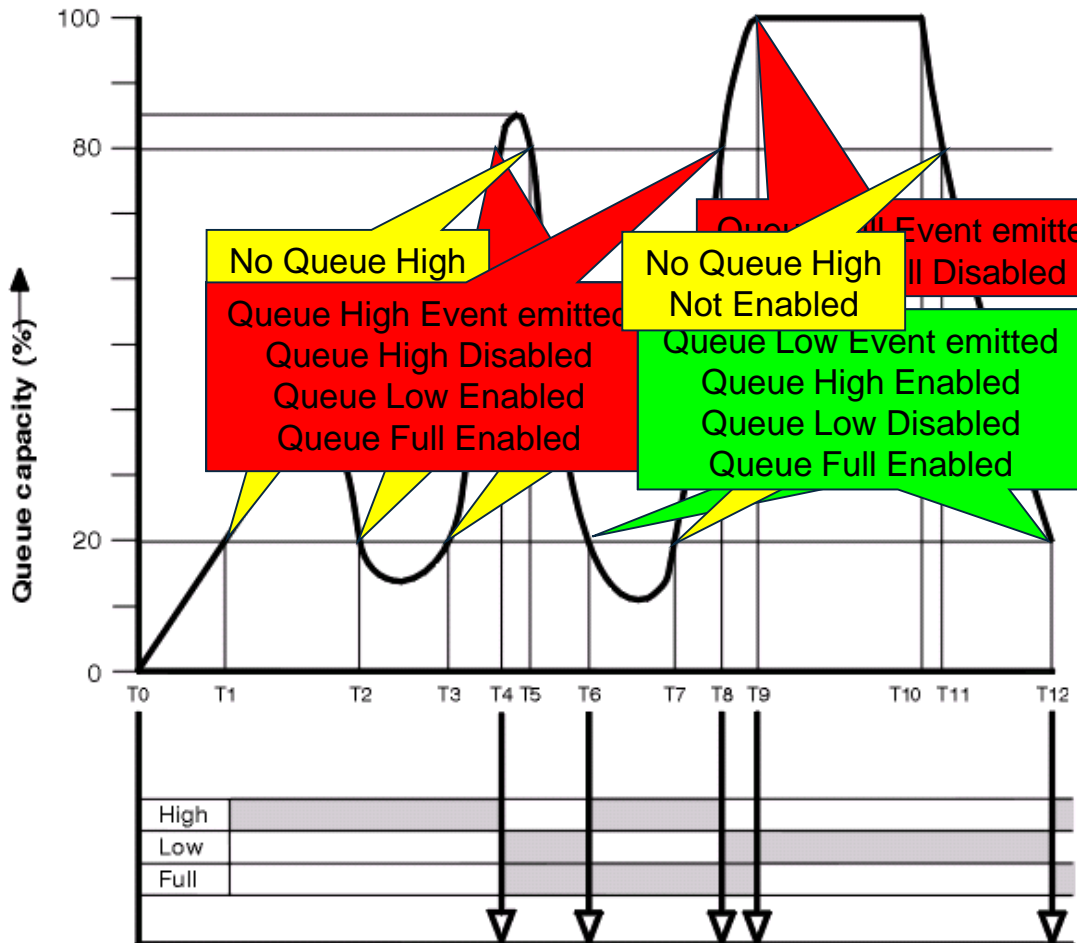
# Performance Event Message Details – Notes

- So this is not the most interesting table we've seen today. All these different event messages have exactly the same content. What is perhaps more interesting is the way these events operate which we will look at on the next page.
- Note that the statistics that are provided as part of a performance event are reset at various points, when a performance event is emitted; when the queue manager restarts; and when a command is issued to reset the statistics. The Time Since Reset parameter in the event message details when the last one of these happens. If no reset has happened it will contain the time of the last event message.





# The Highs and Lows of Performance Events



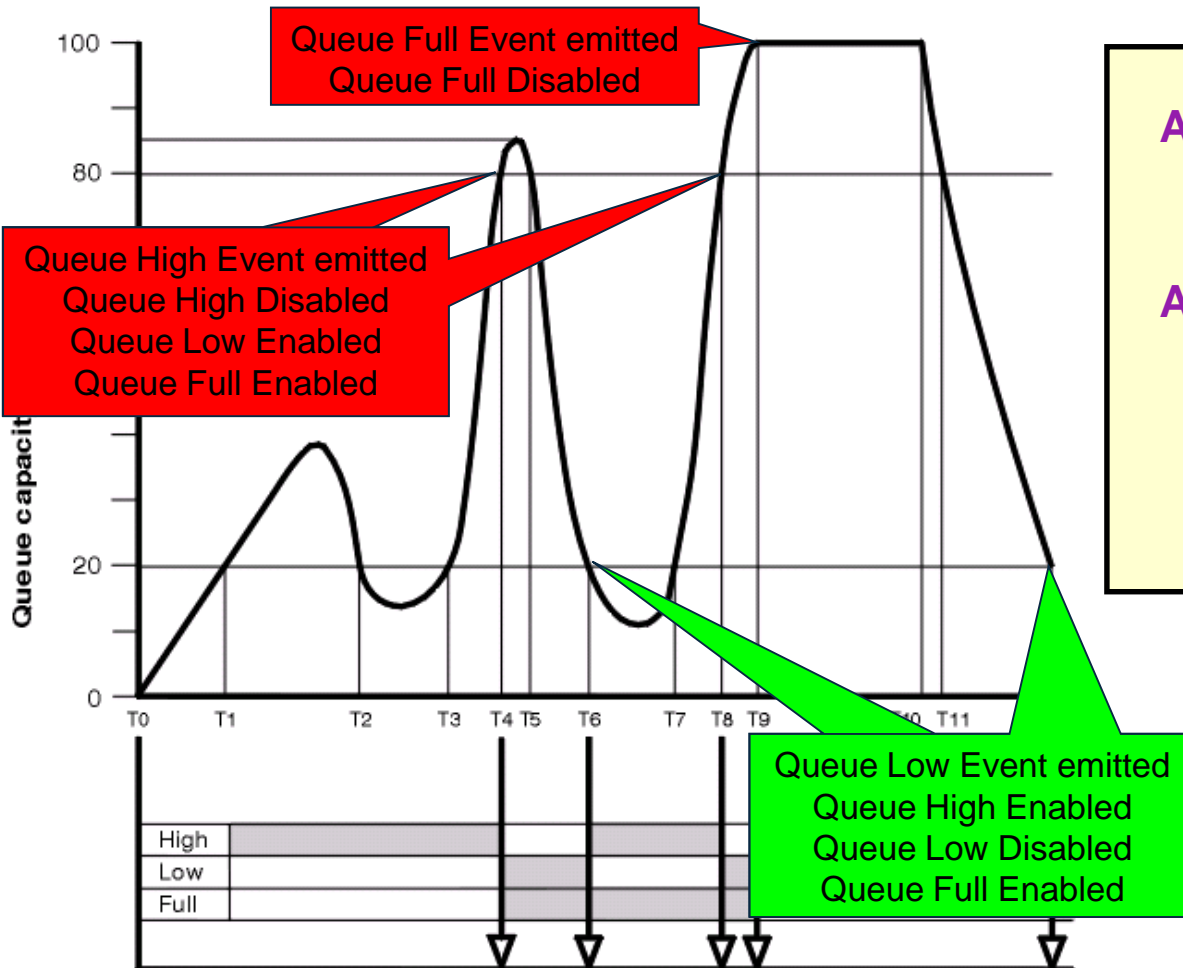
```

ALTER QMGR
PERFMENV(ENABLED)

ALTER QLOCAL(q-name)
MAXDEPTH(1000)
QDEPTHHI(80)
QDEPTHLO(20)
QDPHIEV(ENABLED)
    
```

Complete your session evaluations online at [www.SHARE.org/Orlando-Eval](http://www.SHARE.org/Orlando-Eval)

# The Highs and Lows of Performance Events



```
ALTER QMGR
PERFMEV(ENABLED)

ALTER QLOCAL(q-name)
MAXDEPTH(1000)
QDEPTHHI(80)
QDEPTHLO(20)
QDPHIEV(ENABLED)
```

Complete your session evaluations online at [www.SHARE.org/Orlando-Eval](http://www.SHARE.org/Orlando-Eval)

# The Highs and Lows of Performance Events - Notes

- Queue Service Interval and Queue Depth events are unusual compared to the other events we have looked at in that they operate in pairs. There is a High event which alerts the monitor of a problem, a slow processing application or backlog of messages, and then when the problem is solved, and the queue is being serviced in a timely manner again, or the depth has returned to a nice low value, the corresponding Low event is emitted.
- This means that only one of the High or Low event from the pair is set at any one time and when the event is generated, let's say the High one, then the High event switch is disabled and the Low event switch is enabled. The reverse happens when the Low event is reached.
- For Depth events there is also the Queue Full event to consider. This is automatically switched on when the Queue High state is reached. There isn't an equivalent event for the Queue Service Interval pair though.



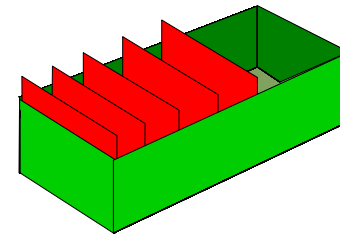
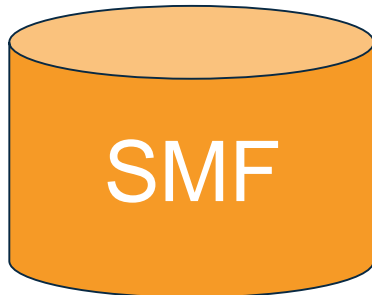
Notes

# Accounting and Statistics

Complete your session evaluations online at [www.SHARE.org/Orlando-Eval](http://www.SHARE.org/Orlando-Eval)

# Accounting and Statistics – Platform differences

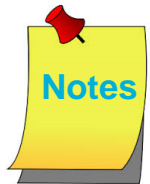
- What is statistics and accounting ?
  - **Statistics:** Data about the activities occurring in the Queue Manager
  - **Accounting:** Data about operations (MQI) performed by applications/channels
  
- z/OS
  - Written as SMF records
  - Type 115 and 116
  
- Distributed
  - Written as PCF format messages
  - On specific queues



SYSTEM.ADMIN.ACCOUNTING.QUEUE  
SYSTEM.ADMIN.STATISTICS.QUEUE

# Platform Differences – Notes

- The z/OS and Distributed platforms both supply the ability to collect Statistics and Accounting Information, but the manner in which they do differs on z/OS to the distributed platforms,
- On z/OS, MQ uses a z/OS component, the System Management Facility (SMF), to collect this information.
- On distributed, no such component is available, so the MQ queue managers write equivalent information onto queues in PCF format (similar to the event messages we have already seen).



Complete your session evaluations online at [www.SHARE.org/Orlando-Eval](http://www.SHARE.org/Orlando-Eval)

# Statistics Records (z/OS)

- SMFSTAT = YES
  - CSQ6SYSP macro
  - SET SYSTEM command
- START TRACE(STAT) DEST(SMF) CLASS(n) STATIME
  - CSQ6SYSP macro
  - SET SYSTEM command
- Provides information about the various parts of the queue manager and channel initiator address spaces.
- SMF 115 Subtype 1
  - System information, for example, related to the logs and storage.
  - Class 1
- SMF 115 Subtype 2
  - Information about number of messages, buffer, and paging information. Queue-sharing group information related to the coupling facility and DB2.
  - Class 2
- SMF 115 Subtype 3
  - More detailed information about storage usage in the MSTR address space.
  - Class 2 and 3
- SMF 115 Subtype 215
  - Bufferpool information when the queue manager is running in OPMODE=(NEWFUNC,800) mode.
  - Class 1
- SMF 115 Subtype 231
  - System information for the CHIN address space.
  - Class 4
  - NEW in IBM MQ V8

# Statistics Records (z/OS) – Notes

- SMF statistics records can start being gathered, either immediately the queue manager (or CHINIT for its statistics) address space is started by using the SMFSTAT=YES parameter in the CSQ6SYSP macro (or equivalent SET command); or when need by issuing the appropriate START TRACE command.
- The STATIME parameter indicates how often SMF records will be cut. By default it is every 30 minutes. A value of zero means that the SMF statistics broadcast of z/OS is used (i.e. using global SMF interval)
- A new SMF record has been added in IBM MQ V8. SMF 115 sub type 231 has the CHINIT control information like adapter and dispatcher task CPU times, DNS resolution times. This helps with tuning the number of tasks configured.
- There are new DSECTs that are shipped that describe these records. CSQDQWSX (QWSX) which is the Self defining section for subtype 231 and CSQDQCCT (QCCT) which is the definition for CHINIT statistics data



Notes



# Accounting Records (z/OS)

- SMFACCT = YES
  - CSQ6SYSP macro
  - SET SYSTEM command
- START TRACE(ACCTG) DEST(SMF) CLASS(n)
- STATIME
  - CSQ6SYSP macro
  - SET SYSTEM command
- Queue Accounting
  - Queue Manager Attribute ACCTQ
  - Queue Attribute ACCTQ
- Channel Statistics
  - Queue Manager Attribute STATCHL and STACLAS
  - Channel Attribute STATCHL
- SMF 116 subtype 0
  - Message manager accounting records
  - Class 1
- SMF 116 subtype 1 & 2
  - Accounting data for each task, at thread and queue level.
  - Class 3
- SMF 116 subtype 10
  - Accounting data for channels
  - Class 4
  - NEW in IBM MQ V8

**ALTER QMGR ACCTQ(ON)  
STATCHL(HIGH)  
STACLAS(QMGR)**

**ALTER QLOCAL(*q-name*) ACCTQ(QMGR)**

**ALTER CHL(*chl-name*) CHLTYPE(SDR)  
STATCHL(QMGR)**

# Accounting Records (z/OS) – Notes

- SMF accounting records can start being gathered, either immediately the queue manager (or CHINIT for channel accounting) address space is started by using the SMFACCT=YES parameter in the CSQ6SYSP macro (or equivalent SET command); or when need by issuing the appropriate START TRACE command.
- The STATIME parameter indicates how often SMF records will be cut. By default it is every 30 minutes. A value of zero means that the SMF statistics broadcast of z/OS is used (i.e. using global SMF interval)
- A new SMF accounting record has been added in IBM MQ V8. SMF 116 sub type 10 has the per channel accounting data like bytes sent, achieved batch size, etc.
- There are new DSECTs that are shipped that describe these records. CSQDQWS5 (QWS5) which is the self defining section for subtype 10 and CSQDQCST (QCST) which is the definition for channel accounting data.
- The granularity of collection of the queue and channel accounting records can be controlled by attributes on queue and channel objects. These can also be configured to refer to a queue manager wide value so that they can easily be turned on and off. For cluster channels, where no object exists locally to configure an attribute, there is a queue manager level setting, STATACLS.

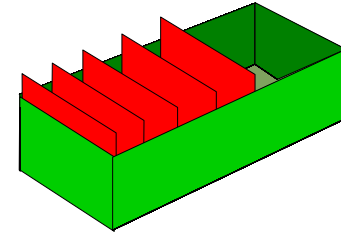


Notes

Complete your session evaluations online at [www.SHARE.org/Orlando-Eval](http://www.SHARE.org/Orlando-Eval)

# Accounting Messages (Distributed)

- Collects information about the applications which connect to the Queue Manager
  - Overall
  - Per Queue
- Queue Manager Attribute ACCTINT
- Overall MQI Accounting
  - Queue Manager Attribute ACCTMQI
- Queue Accounting
  - Queue Manager Attribute ACCTQ
  - Queue Attribute ACCTQ
- PCF Header fields
  - Command
    - MQCMD\_ACCOUNTING\_MQI – overall information
    - MQCMD\_ACCOUNTING\_Q – queues message



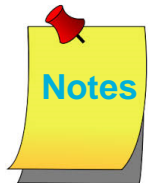
**SYSTEM.ADMIN.ACCOUNTING.QUEUE**

```
ALTER QMGR
      ACCTINT(1800)
      ACCTMQI(ON)
      ACCTQ(ON)
```

```
ALTER QLOCAL(q-name)
      ACCTQ(QMGR)
```

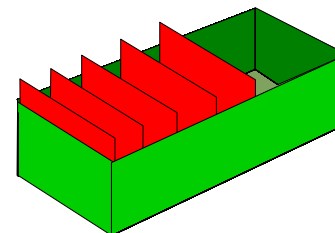
# Accounting Messages (Distributed) – Notes

- Accounting Monitoring Data collects information about the applications which connect to the Queue Manager. This data, when enabled by means of the ACCTMQI switch on ALTER QMGR, is written in the form of PCF records to the SYSTEM.ADMIN.ACCOUNTING.QUEUE and can be post-processed to give information on the activity of each application connected. The PCF formatted message is written upon MQDISC (or at regular intervals for long running tasks, controlled by the ACCTINT attribute on ALTER QMGR).
- As well as the standard collection details, detailed queue information for each connection may also be collected. Written at the same time as the standard accounting data collection, a number of PCF formatted messages may be written which includes accounting information for each queue opened by the connection (up to 100 queue details per message). The queues that this information is written for is controlled by the ACCTQ attribute on queues and on the queue manager (providing a way to switch on and off for multiple queues at once).
- The PCF Header of an accounting data message will record the Command field as MQCMD\_ACCOUNTING\_MQI for the main accounting information and MQCMD\_ACCOUNTING\_Q for the additional messages written for queues.



# Statistics Messages (Distributed)

- Collects information about IBM MQ resources
  - Queue Manager
  - Per Queue
  - Per Channel
- Queue Manager Attribute STATINT
- Overall MQI Statistics
  - Queue Manager Attribute STATMQI
- Queue and Channel Statistics
  - Queue Manager Attributes STATQ, STATCHL and STATACLS
  - Queue Attribute STATQ
  - Channel Attribute STATCHL
- PCF Header fields
  - Command
    - MQCMD\_STATISTICS\_MQI
    - MQCMD\_STATISTICS\_Q
    - MQCMD\_STATISTICS\_CHANNEL



**SYSTEM.ADMIN.STATISTICS.QUEUE**

```

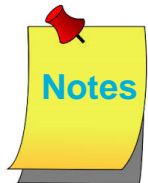
ALTER QMGR STATINT(1800)
      STATMQI(ON) STATQ(ON)
      STATCHL(HIGH)
      STATACLS(QMGR)

ALTER QLOCAL(q-name)
      STATQ(QMGR)

ALTER CHL(chl-name)
      CHLTYPE(SDR)
      STATCHL(QMGR)
  
```

# Statistics Messages (Distributed) – Notes

- Statistics Monitoring Data collects information about IBM MQ resources and writes this information at configured intervals (using the STATINT attribute on ALTER QMGR) to the SYSTEM.ADMIN.STATISTICS.QUEUE. This data, written in the form of PCF records, can be post-processed to give information on the activity of the system.
- Data collection is split into 3 classes, statistics based on the activity of the whole system; statistics on the activity of the queue (per queue); and statistics on the activity of the channel (per channel).
- The PCF Header of an accounting data message will record the Command field as MQCMD\_STATISTICS\_MQI for the activity of the whole system; MQCMD\_STATISTICS\_Q for the per queue activity and MQCMD\_STATISTICS\_CHANNEL for the per channel activity.

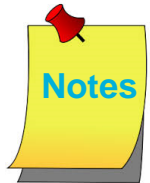


# What can I discover?

- Open details
  - first open time, last close time for a queue
  - MQCMD\_ACCOUNTING\_Q message on Distributed
  - SMF 116 subtype 1 WQ block
- MQGET details
  - count, total bytes, msg-size (min / max), time-on-queue (min / avg / max) (persistent / non-persistent) for a queue
  - MQCMD\_ACCOUNTING\_Q message on Distributed
  - SMF 116 subtype 1 WQ block
- API counts
  - for each of the MQ API's (MQPUT / MQGET/ ...) across the queue manager
  - MQCMD\_STATISTICS\_MQI on Distributed
  - SMF 115 Subtype 2 CSQDQMST – Message manager
- Number of messages transferred (persistent / non-persistent) on a channel
  - MQCMD\_STATISTICS\_CHANNEL on Distributed
  - SMF 116 Subtype 231
- Batch Info (total / number full / avg size) on a channel
  - MQCMD\_STATISTICS\_CHANNEL on Distributed
  - SMF 116 Subtype 231

# What can I discover? - Notes

- There is lots of information to be discovered in your accounting and statistics records. We could spend a whole hour just looking at these! So instead here are some examples of the sort of information you can get out of them on both the z/OS and Distributed platforms.



Complete your session evaluations online at [www.SHARE.org/Orlando-Eval](http://www.SHARE.org/Orlando-Eval)



# Accounting Message Details

## MQCMD\_ACCOUNTING\_MQI

Connection Details (application name / process Id / connection type / connect time)

API counts (MQOPEN / MQCLOSE / MQGET / MQPUT / ...)

MQPUT / MQGET / MQGET (browse)

message counts: persistent / non-persistent

bytes counts: persistent / non-persistent

message sizes: persistent / non-persistent

## MQCMD\_ACCOUNTING\_Q

Queue details: name, type

Open details: first open time, last close time

MQPUT details: count, total bytes, msg-sizes (min / max) (persistent / non-persistent)

MQGET details: count, total bytes, msg-size (min / max), time-on-queue (min / avg / max) (persistent / non-persistent)

MQGET(browse): count, total bytes, msg-sizes (min / max) (persistent / non-persistent)

# Statistics Message Details

## MQCMD\_STATISTICS\_MQI

API counts for each of the MQ API's (MQCONN / MQDISC / MQPUT / MQGET/ ...)

Total message/bytes put to queues (persistent / non-persistent messages)

Total message/bytes got from queues (persistent / non-persistent messages)

## MQCMD\_STATISTICS\_Q

Minimum / Maximum depth of queue

Average time-on-queue for messages retrieved from the queue

API counts for GET / PUT / BRWS (persistent / non-persistent)

Total byte counts for GET / PUT / BRWS (persistent / non-persistent)

## MQCMD\_STATISTICS\_CHANNEL

Number of messages transferred (persistent / non-persistent)

Number of bytes transferred (persistent / non-persistent)

Network times (min / avg / max) : measured on heartbeats

Exit Times (min / avg / max)

Batch Info (total / number full / avg size)

Number of PUT retries

# Any Questions ?



**Please complete your session evaluation .. Thank You !**

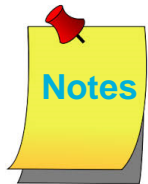


Complete your session evaluations online at [www.SHARE.org/Orlando-Eval](http://www.SHARE.org/Orlando-Eval)

# Copyright and Trademarks

© IBM Corporation 2015. All Rights Reserved.

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).



Complete your session evaluations online at [www.SHARE.org/Orlando-Eval](http://www.SHARE.org/Orlando-Eval)