Take Note!

Before using this Manual and the product it supports, be sure to read the general information under "Notices".


This edition applies to Version 1.0 of MA0B: MQSeries Knowledge Base and to all subsequent releases and modifications until otherwise indicated in new editions.

A form for reader's comments is provided at the back of this publication. If the form has been removed, address your comments to:

IBM United Kingdom Laboratories
Transaction Systems Marketing Support (MP187)
Hursley Park
Hursley
Hampshire, SO21 2JN, England

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you. You may continue to use the information that you supply.

© Copyright International Business Machines Corporation 1997. All rights reserved.
Note to U.S. Government Users - Documentation related to restricted rights - Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corporation.
# Contents

## Introduction
- A Welcome to MQSeries Knowledge Base ................................. 1

## CICS
- Dynamic Queues ................................................................. 2
- MQCALL Open Queue failure on MVS ...................................... 3
- MQSeries MVS 1.2 and CKTI ............................................... 3
- Sharing An Initiation Queue Amongst Multiple CICS Regions ............. 4
- Channel initiator on MVS ..................................................... 4
- CICS shutdown normally ..................................................... 5
- MQ/ESA can one MQ connect to more than 1 CICS? ......................... 6
- Client attachment Feature .................................................. 6
- COD, Confirmation of Delivery.................................................. 6
- MARK_SKIP_BACKOUT behaviour ........................................... 7
- MQI and CICS ................................................................. 8
- CICS Message Exit Receive End pointer definitions ....................... 9
- Triggering in CICS .......................................................... 11

## Client
- MQ and Java ................................................................. 13
- MQLink for Notes ............................................................ 13
- MQI channel sharing .......................................................... 14
- MQCONN error AMQ9202 .................................................. 14
- MQSeries Win95 Client supports Windows 98 .............................. 15
- MQ Client for Java ........................................................... 15
- Lotus Notes Remote Agent .................................................. 15
- Question on AccessQueue method of MQLSX ................................ 16

## Data Conversion
- Accented chars between OS/2 and AIX ...................................... 17
- Data conversion across platforms ......................................... 17
- EBCDIC to ASCII Conversion Questions .................................... 18
- MQEI Data Conversion ......................................................... 19
- MVS to OS/2 Data conversion ............................................... 20
- Big Endian / Little Endian ................................................... 21
- CCSID on AIX and HP .......................................................... 22
- CCSID on MQ .................................................................. 23
- Coded character set conversions ........................................... 24
- CORRELID and MSGID ........................................................ 25
- Data Conversion OS/2 - MVS ............................................... 27
- Questions Ref: Clients & Sender channels................................. 28

## General
- Authority for RUNMQSC commands ......................................... 30
- Basic MQSeries concepts .................................................... 30
- Basic questions ............................................................... 30
- Channel Connection Name ................................................... 31
- Deleting expired msgs from xmitq ........................................ 32
- Events Reading ............................................................... 33
- Message Priority ............................................................. 33
MQ in comparison to RPC and CICS .......................... 34
MQS administration interface .............................................. 35
MQSeries List Server .................................................. 35
MQSERIES PROCESS NAME & DESCRIPTION ...................... 36
Question about when Messages Expire ................................. 38
Use of datagrams and replytoQ ............................................. 39
Using MQI to monitor and controlling channels .............................. 40
When do you really need MQ Series "Client" component vs server? .......................... 40
XMIT Q truncation feature ............................................... 41
EDI and MQSeries ................................................... 42
"MQ Directory" - MQSeries resources on the WWW ...................... 42
Are transmission queues safe during a failure ................................. 43
Queue Architecture .................................................. 43
Can different platform processes send messages to the same Q? .......................... 46
Channel initiator for OS2 and AIX ........................................ 47
CHL types for Switched Lines ............................................. 48
COMMANDS from an Application Program ................................ 48
Syncpoint and Commit .................................................. 49
Cross Platform communications - creating and sending messages ........................ 49
Dead Letter Queue ................................................... 49
Dead Letter Queue ................................................... 50
Default persistence when using ReplyToQ .................................. 51
Default transmission Queue ............................................. 51
Design question ..................................................... 52
Exact size of the message header ........................................... 53
EXPIRY ......................................................... 54
Full Xmit queue ..................................................... 54
Message channels and multiple connections ................................. 54
Message Prioritization ................................................ 55
MQGMO_BROWSE_NEXT sometimes does not process a message .......................... 56
MQI channels vs. Message channels ........................................ 57
MQI Client MQCONN returns code 2059 .................................. 57
Not a Valid Trigger Message ............................................. 58
Performance Tuning .................................................. 58
Priority of event messages ............................................... 59
Priority of messages .................................................. 59
PTF Download ..................................................... 60
Queue Manager Hierarchies for enterprise networks ................. 60
Queue priority ..................................................... 61
Remote Q and the DeadLetter Q ........................................... 62
Rules of thumb ..................................................... 62
runmqchl on AIX .................................................... 63
Security Exit ..................................................... 64
Self Study Guide .................................................... 66
Sender/Receiver Question ............................................. 68
The length of Msg .................................................. 68

Duplicate reason codes in MQ V5 ........................................ 70
Question ..................................................... 70
Technique to keep channel up ............................................ 71
Trigger Interval Value .................................................. 72
Trigger monitor for MVS ................................................ 72
Trigger on last message .................................................. 73
Use of trigger queues, initiation messages and CSQQTRMN ................. 74
MQ/NT C++ Compier .......................................................... 75
OS/2 queue manager access .................................................. 76

IMS .......................................................................................... 78
MQM 1.1.4 IMS Bridge: How can we prevent flooding IMS Q's? .... 78
MQSeries IMS-Bridge with messages > 32K ............................. 80
Accessing MQ IMS-Bridge from MQSeries Java ....................... 83
IMS BMP simultaneously connected to multiple queue managers 83
IMS Bridge and CommitMode .................................................. 84
IMS BRIDGE: Local or MONOPLEX MVS System .................... 85
Installing the IMS Bridge ........................................................ 85
MQ and OTMA Manuals .......................................................... 86

INTEL ...................................................................................... 88
MQ 2.0 for NT - Domain user vs. local user .............................. 88
Authorization to put reports on queue using MQSeries for NT v5 88
Installation of Windows NT Version ........................................... 89
MQ NT Qmgr Disconnect on login ............................................ 90
MQ/NT & Multithreading questions .......................................... 91
MQSERIES for WIN/NT Setup Failure. ................................. 91
Permission problem with MQSeries for Windows NT ................. 92
MQSeries for Windows NT on a Backup Domain Controller .... 93
MQSeries for Windows NT and Local/Domain logon problem .... 93
MQ Series Backup Failure ...................................................... 94
Authorization for OS/2 ........................................................... 94
OS/2 high depth and low depth settings ................................... 95

MVS .......................................................................................... 96
Allocation of MQM/ESA LOG dataset, LOG archive. ............... 96
Channel indoubt ................................................................. 97
Channel initiator failure w/OS390 1.3 ...................................... 97
DMQ with and without CICS .................................................. 98
INTERTEST and MQSERIES on MVS ................................. 99
Persistent Messages MVS/ESA system. ................................. 99
MQSeries ISPF panels ........................................................... 100
MQSeries/ESA V1R2 timestamp used ..................................... 100
MVS Listener ................................................................. 101
Open Overhead using DQM in MVS .................................... 101
Reading the message descriptor from MVS ........................... 102
Decrease size of a pageset in MVS/ESA ................................. 102
Dual Logging versus DASD Dual Write ................................. 104
Handling of non-persistent messages in MVS ......................... 105
MQ APPC security ............................................................. 106
MQ Clients to MVS ............................................................. 107
MQ Subsystem objects on a sysplex ...................................... 108
MVS Performance Considerations ......................................... 109
Queue Manager Name .......................................................... 110
Using MQ to initiate a batch job ............................................ 110

Unix ...................................................................................... 111
AMQ8145 Connection broken / HP-UX ................................. 111
MQSeries V5 now supports HP-UX V11 ............................... 111
Error AMQ9523 and AMQ9207 in LOG ............................... 112
MQ on AIX ................................................................. 113
N0 Default QManager (AIX) ............................................. 113
Permission problem .............................................................. 114
AIX MQseries QA ............................................................. 114
AIX SIGALRM restriction .................................................... 115
Kernel Configuration on Sun Solaris 2.2 .............................. 115
MQSeries for AIX and CCSID ............................................. 116
MQSeries for AIX and DCE threads ....................................... 116
Questions on MQ recovery ................................................... 117
TPname for AIX to MVS using LU 6.2 ................................. 118
Trigger on HP-UX ............................................................. 118
Notices.

The following paragraph does not apply in any country where such provisions are inconsistent with local law.

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates.

Any reference to an IBM licensed program or other IBM product in this publication is not intended to state or imply that only IBM's program or other product may be used. Any functionally equivalent program that does not infringe any of the intellectual property rights may be used instead of the IBM product. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, New York 10594, USA.

The information contained in this document has not be submitted to any formal IBM test and is distributed AS IS. The use of the information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item has been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

- AIX
- Anynet
- CICS
- ESA
- IBM
- MQSeries
- MVS
- OS/2
- RISC 6000
- SNA
- System/390
- TXSeries
- VisualAge

The following terms are trademarks of other companies:

- INTEL, Intel Corporation
- Lotus, Lotus Corporation
- NT, Windows, Microsoft Corporation
## Summary of Changes

<table>
<thead>
<tr>
<th>Date</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>18th August 1998</td>
<td>Initial release</td>
</tr>
</tbody>
</table>
Preface

The material in this document contains information extracted from internal IBM employee as well as external customer conferences and is intended to provide a quick and convenient source of technical information concerning MQSeries.
Introduction

A Welcome to MQSeries Knowledge Base

Welcome to MQSeries and the subject of IBM's Messaging and Queuing Interface (the MQI).

Messaging and Queuing was introduced by IBM in September 1992 as a fundamental part of the Networking Blueprint for the future. This communications style is widely recognized by IBM and the communications industry in general as one of the three fundamental paradigms for application to application communications (the other two being Conversations and Remote Procedure Calls).

The MQI allows improved programmer productivity by providing a simple, flexible and reliable means of building distributed applications. Communication between programs is critically important in business today; yet there are many obstacles and contingencies to be considered. For example, it is usually necessary for a live connection to be maintained between programs, but if any element that mediates the connection fails, the communication stops. The MQI, which provides access to Messaging and Queuing services, enables programs to send messages to applications that are not connected or active. Thus a major differential benefit of the MQI is the offering of time independence to application programmers in sending and receiving data between applications.

The information in this MQSeries Knowledge Base is classified primarily by platform. General use questions are in their own category.
Question

We're trying to create a simple application which sends some data to CICS and gets it back on a dynamic queue. We have two problems:

Something on the CICS side is not letting MQSeries start the transaction. Our support people seem to be able to start things up eventually, but we'd like to know what's going on and we're only in indirect communication with them. What could be happening?

The second problem is the return data. The data gets back to the sending system, but is put on the dead letter queue. I look at the dead letter header and it matches the queue name generated by the OPEN of the model queue. I even look at the queues defined to MQSeries while the program is running and I see the dynamic queue name. What can I look for to figure out why the data is not being put on the dynamic queue? Thanks,

Answer

1. CICS transaction initiation

I think that first of all you need to find out how they have set up the transaction triggering, and confirm that they have a Trigger Monitor ACTIVE on the appropriate initiation Queue. Further to that you will need to investigate the Trigger Setting on your target queue, and ensure that it has not been set OFF for some nefarious reason.....

2. Msg on DLQ.... look at the DLQ header structure... there will be a REASON CODE at offset 8 (I think) which will tell you WHY.. Has the Model Queue that you opened got the correct attributes to permit the MCA to put msgs on it while you (presumably) have it OPEN with an outstanding GET WAIT.

Hope this gives you something to go forward with.

Resolution

Thanks for the input. Problem one was caused by our not having the CICS transaction defined at the time CKTI tried to start it. If it failed once, you have to do some magic to get it started again. Our transaction is now autoinstalled, so this should no longer happen.

Second problem was twofold. Dynamic queue not properly defined to allow Put by MQS while opened for get by application AND host application trying to put a persistent message (evidentially, not allowed to dynamic queues). Thanks for all the help.

Answer

Good.. Glad you've got it fixed. To clarify: Yes you CANNOT put a persistent msg to a TEMP DYN Q. You can however to a Perm Dyn Queue.
MQCALL Open Queue failure on MVS

Question

We have a CICS application that is trying to open a queue. The program abends with an abend code of QAPI in module CSQCTRUE. Any suggestion as to what causes this?

Answer

The QAPI abend code means that your program has issued an unrecognizable MQSeries API call.

Only the following MQSeries API calls will be processed by the Task Related User Exit of the CICS adapter:

MQCLOSE
MQGET
MQINQ
MQOPEN
MQPUT
MQPUT1
MQSET

MQCONN, MQDISC, MQBACK and MQCMIT are also handled but before the TRUE processing. Any other call will produce a QAPI abend.

MQSeries MVS 1.2 and CKTI

We are running MQseries MVS 1.2 and are using trigger monitor in CICS environment (CICS/ESA 4.1).

We successful started CKTI transaction and were able to trigger CICS task that empties Message queue, but after a variable time (12-15 minutes) CKTI task is abended with AEXY code.

CICS definitions for CKTI transid and CSQC* programs are the MQ supplied ones.

Has anyone experienced such a problem with MQSeries/MVS 1.2?

Answer

The AEXY abend means that CKTI has been resumed but was purged during its wait.

The questions to ask are who tried to purge the CKTI transaction and why?

Response

We discovered a user program that is purging all tasks that are waiting over a specified interval time. The customer is excluding CKTI from their program processing.
Sharing An Initiation Queue Amongst Multiple CICS Regions

Question

Is it possible to share an initiation queue among multiple CICS regions?

Our situation is as follows:

- Customer has multiple CICS regions, each running the same set of applications, but divided to service users in different geographical regions, to disperse the CICS workload.
- CICS application A puts a msg into Q1, which in turn triggers another CICS application B, which process the msg. Q1 is defined to use INITQ1 for its INITQ.
- A and B are running in all CICS regions.
- So if INITQ1 cannot be shared by the CKTIs in the multiple CICS regions, i.e., if we have to define different INIT Qs for each region, it means we have to define different application Qs as well. And then the application has to be able to use different application Qs depending on which region it is running in. Not a pretty task...

In the MQ/MVS System Mgmt Guide, it states "You cannot start more than one instance of CKTI against the same initiation queue", so I'm a bit worried. Does this statement apply to multiple CKTIs on different CICS regions also?

Answer

To put your mind at rest the statement you refer to is talking about a single CICS system.

There is nothing to stop you starting one CKTI in each CICS system, connected to a queue manager, against the same initiation queue.

When a trigger message does arrive on the initiation queue it is not possible to determine which CKTI's MQGET will be processed.

Sharing the queue

Question

I have 2 application running at the same time. One put messages to a queue and other one reads. It looks like they are not sharing the queue though I set share option. Is it possible to share the same queue by both application? If so, what are all options I need to specify?

Answer

No particular options are required for one process to be putting and one process getting from a queue at any one time. If you are observing that the getting process does not get anything until the putting process terminates, the most likely explanation is that you are doing the puts in syncpoint (the default on MVS). In that case the getting process will not "see" the messages until the putting process commits them. Disconnecting from the queue is an implicit commit.
Question

Let me pose a variation to the original question:

In a CICS environment, suppose there are multiple copies of the same application reading from a queue. Does the first copy to start have an exclusive read on the queue until it finishes (at which time the next copy can start reading) or can multiple transactions read from one queue "simultaneously"?

Answer

Assuming the queue has been opened for shared input, multiple transactions can read the queue at the same time. For normal non-browsing MQGET's the Qmgr serializes the MQGET's so each transaction instance will get a different message.

Channel initiator on MVS

Question

Is it possible to have more than 1 (one) channel initiator running on MVS?

Answer

On a single MVS image you can have multiple Queue managers each of which may have at most one channel initiator. This means that a single queue manager may not have multiple channel initiators.

CICS shutdown normally

Question

When running with MQSeries, we have 3 CICS transactions that are always running. For example cksg and ckti. We know how to stop cksq by disconnecting the channel, and for ckti we issue the operator command ckqc stopckt. Now if we knew how to stop ckam, we should then be able to shutdown CICS normally. We are now purging ckam and them doing a CICS perform shutdown. Can you tell me of a way to stop ckam, without purging it?
**Answer**

You can not stop CKAM. CKAM will terminate itself when all pending events have expired.

If you have only tried to connect to one named queue manager and have subsequently disconnected successfully then CKAM will terminate.

Pending events are 'deferred connection' and 'termination notification'. If any are outstanding then CKAM will stay active.

Page 134 of the *System Management Guide* gives more details.

---

**MQ/ESA can one MQ connect to more than 1 CICS?**

**Question**

We are new site installing MQ/ESA. We have multiple CICSs for different development and testing stages. Can all the CICSs connected to one MQ started task, or each CICS has to have its own MQ started task?

**Answer**

Each CICS system may connect to only one MVS queue manager. However multiple CICS systems can connect to the same MVS queue manager.

---

**Client attachment Feature**

**Question**

Where I one get details on the Client Attachment Feature for MQSeries MVS/ESA R114?

**Answer**

The *MQSeries Clients* publication GC33-1632 is an excellent introduction, overview and guide. MVS Client support, once the CAF feature has been installed, is the same as for other server platforms. For details on how to configure the server, refer to the *Systems Management Guide for MVS*, SC33-0806.
Question

If an MQ application puts a message to a remote queue with COD, and at the remote end a CICS application gets the message from that queue but abends. Will the message be "backed-out" and put back on the queue?

Answer

The simple answer is "It depends". Provided that you get the message with syncpoint control (ie, use MQGMO_SYNCPOINT) then your answer is "Yes". If you do NOT use syncpoint control then your answer is "No".

There is a good section about this in the MQSeries for MVS/ESA Application Programming Reference manual, in the section dealing with the MQMD. In the 1.1.3 manual look at the heading MQRO_COD at the foot of page 31. However, in case you don't have a copy of the manual to hand:

The "Yes" case is as follows:

The message with COD set is got by CICS in-syncpoint. Because it is got in-syncpoint, the COD message which is generated for you at this time is put in-syncpoint.

Then, when your CICS task ABENDs, BOTH messages are backed out, meaning that the COD message is "discarded", and the original message is reinstated.

If your CICS task commits, then the reverse happens, and your COD message is committed to the queue, and the deletion of the original message is committed.

The "No" case is:

You get the original message out of syncpoint, so MQSeries immediately generates your COD message out of syncpoint. Both the original message get and the COD message put are committed at that point, irrespective of what later happens to your CICS task.

**MARK_SKIP_BACKOUT behaviour**

Question

In what documentation is the behaviour of MQGMO_MARK_SKIP_BACKOUT described? All I find references this as MQGET GMO options - that a message get with this option will not be rolled back by a CICS SYNCPOINT ROLLBACK. Only a SYNPOINT will destroy this message. That's all information I can find. The situation in our context is the following:

1. We get the message with MARK_SKIP_BACKOUT + SYNCPOINT
2. We call a 'user program' which does a ROLLBACK
3. The program returns to us
4. We detect that the 'user program' did a ROLLBACK but due to the fact that we don't know whether this program did some other updates to some resources, we do an additional SYNCPOINT ROLLBACK. After some cleanup, we're doing a SYNCPOINT (Commit).
What we experience is that after the second ROLLBACK the message is put back to the input queue. That's the reason why we get restarted until we detect a poisoned message!! That's not what we intended to have. Is this behavior documented somewhere or is it a bug in MQ?

We're using CICS 3.3 and MQM 114.

**Answer**

There's a pretty good description of the operation of MARK_SKIP_BACKOUT Application Programming Guide, Chapter 10, "Skipping backout".

From a brief examination of what you describe, everything seems to be working as described in the manual.

To fix your program, I think you need to add logic to step 4 along the lines:

```c
if called program has issued backout, then
  do;
    MQPUT(Poison_message, Poison_queue)
    SYNCPOINT
  end;
```

This will remove the poison message from future input processing.

The main thing to remember is that it takes two ROLLBACKs to remove the MARK_SKIP_BACKOUT "backwards", but only one COMMIT to remove it "forwards". This is because in the case of ROLLBACK of a MARK_SKIP_BACKOUT message, (in essence) a new unit of work is started in which the MARKed message is the first element.

---

**MQI and CICS**

**Question**

If my application consist of only CICS, are there any reasons why I should use MQI and not TD started transactions?

**Answer**

If you are truly a CICS ONLY installation - i.e. you have no requirements for ANY data access other than between CICS applications, then it is hard to make a case for an additional (chargeable) program offering like MQ.

However, most installations will have some batch jobs at least running and MQ allows Queue sharing between ANY MQ process running in MVS. For example, although TDs can be extrapartition, MQ allows 2-way communication, so as soon as ANY process COMMITs, then subject to access security, the message data is available to ANY MQ MVS process.

A number of customers find this ability for 'instantaneous' access within CICS to data being generated outside of it, extremely useful. In addition MQ's performance (the data is held in bufferpools rather than being directly written to DASD), will probably provide greater OVERALL throughput compared to TD.
MQ provides all the facilities of TD/TS with additional API functions such as:

- the ability to request an ECB to be posted when a particular message arrives
- the ability to - within the scope of a single unit of work to either get or put a message OUTSIDE of that unit of work
- the ability to request various MQ generated reports (COA/COD, expiry) to be sent or requested ReplytoQs
- the ability to create and use dynamic queues extended triggering capability
- the ability to have essentially unlimited Queue depths and a message size up to 4 MB.

All this plus full UOW integrity and recoverability.

In short, MQ provides a much richer API which in itself, could allow more complex CICS transactions to be created, but (I believe) more importantly, allows CICS transactions to share data with other non-CICS processes.

---

**CICS Message Exit Receive End pointer definitions**

**Question**

I have looked through the MQSERIES Distributed Queue Management Guide for information on how to code the DFHCOMMAREA for a CICS program that will be linked to by the Channel Agent. There seems to be very little actual implementation information on how to code the DFHCOMMAREA data definitions in COBOL. It says in the manual that the exit is invoked with the parameters passed as pointers (addresses). Does anyone have any guidance on how this is actually coded in the CICS exit program?

It seems to me that there must be some examples or samples on how to code this.

**Answer**

Here is an example of code extracted from a CICS channel exit program written in COBOL II. Included is the relevant code from the linkage section and procedure division.
LINKAGE SECTION.

*  
*  POINTERS PASSED BY THE MCA  
*  
01 DFHCOMMAREA.
   03 LS-CXP-PTR  POINTER.
   03 LS-CD-PTR  POINTER.
   03 LS-DL-PTR  POINTER.
   03 LS-ABL-PTR  POINTER.
   03 LS-AB-PTR  POINTER.
   03 LS-EBL-PTR  POINTER.
   03 LS-EB-PTR  POINTER.

*  
*  CHANNEL EXIT PARAMETERS  
*  
01 LS-CXPB.
COPY CMQCXPL.

*  
*  CHANNEL DEFINITION  
*  
01 LS-CHANNEL-DEF.
COPY CMQCDL.

*  
*  TRANSMISSION DATA LENGTH  
*  
01 LS-DATA-LENGTH  PIC S9(9) COMP.

*  
*  AGENT BUFFER LENGTH  
*  
01 LS-AGENT-LENGTH  PIC S9(9) COMP.

*  
*  AGENT BUFFER  
*  
01 LS-AGENT-BUFFER.
   03 LS-AGENT-BUFFER-C  PIC X OCCURS 1 TO 4194732
   DEPENDING ON LS-AGENT-LENGTH.
* EXIT BUFFER LENGTH
* WE DON'T USE EXIT BUFFER, AS WE'RE NOT INCREASING
* THE SIZE OF THE MESSAGE WE'VE BEEN PASSED. IT CAN BE
* USED IF WE WANT TO GETMAIN OUR OWN STORAGE AND USE
* THAT INSTEAD OF THE AGENT BUFFER PASSED BY THE MCA

01 LS-EXIT-LENGTH PIC S9(9) COMP.
* EXIT BUFFER
* EXIT BUFFER
01 LS-EXIT-BUFFER.
   03 LS-EXIT-BUFFER-C PIC X OCCURS 1 TO 4194732
      DEPENDING ON LS-EXIT-LENGTH.
PROCEDURE DIVISION.
   SKIP3
* A-MAIN SECTION.
***************
   A-0001.
* ADDRESS FIELDS PASSED BY MCA
* SET ADDRESS OF LS-CXPB TO LS-CXP-PTR.
SET ADDRESS OF LS-CHANNEL-DEF TO LS-CD-PTR.
SET ADDRESS OF LS-DATA-LENGTH TO LS-DL-PTR.
SET ADDRESS OF LS-AGENT-LENGTH TO LS-ABL-PTR.
SET ADDRESS OF LS-AGENT-BUFFER TO LS-AB-PTR.
SET ADDRESS OF LS-EXIT-LENGTH TO LS-EBL-PTR.
SET ADDRESS OF LS-EXIT-BUFFER TO LS-EB-PTR.

Triggering in CICS

Question
I have a customer who has an application that writes to a queue which triggers a transaction in CICS. The queue became full and was writing to the dead letter queue. Upon further research it was found that the trigger on the queue was set to off. I could not tell at what point the queue became full and when the trigger was turned off. Is there anything in MQ that would set the trigger to off. If it is in a problem situation why does it not set it back on when the problem has been corrected.

Answer
This not a problem of CICS-environment. Triggering has a lot of rules which had to be satisfied. If only one rule mismatched, the triggering stops at once. The triggering rules are exactly described in Application Programming Guide (Appendix A).

There are 3 ways that triggering would be switched off for an application queue (ie. queue set to NOTRIGGER):

1. If you have explicitly done it via MQSET or DEFINE/ALTER queue.
2. If the queue is a transmission queue and the channel serving that xmitq has stopped.

3. If the trigger type is depth, then triggering is disabled after the trigger message is generated. This is documented in the Application Programming Guide chapter on triggering.

I suspect 3 might be the problem in your case.

The APG lists all the conditions for triggering and it is possible that one or more of the other conditions is also no longer being satisfied. This would prevent triggering; however, just because one of these other conditions is not satisfied does not mean your application queue would be set to NOTRIGGER. This only happens for the reasons above.

**Response**

In our case it looks like the option 2 was our problem.
**Client**

**MQ and Java**

Can someone give me a thumbnail sketch of the MQ Java architecture.

Are the MQJava classes pure java? If so, how do they access the MQ implementation? Via a private protocol to an MQ node? Is a MQJava client like a normal MQ client in this regard or is the private protocol different for Java clients?

**Answer**

The MQ Java client is an MQ client. As far as the supporting Qmgr is concerned it is just another MQ client. The MQ Java client is written in Java, is imported into your applet as a package, and uses the standard Java TCP/IP classes under the covers to talk to the supporting Qmgr (using the MQ client protocol). Note that Netscape browsers only permit an applet to establish a sockets connection with the same site the applet was downloaded from. The Java client opens a socket to port 1414 on the machine where the MQServer is located. (The actual port is configurable, but you still need some port). This might have firewall implications. Also, because of Netscapes security model, the MQServer must be on the same machine as the web server that served up the applet.

**MQLink for Notes**

**Question**

There are three phases for MQLink for Notes. And the Phase 3 is LSX, Does anyone know whether the Phase 3 has all the functions that Phase 1 and Phase 2 have. If not, In what kind of situation I should use Phase 1 or Phase 2?

**Answer**

**Phase 1:** Notes Client uses Forms with mail-enabled fields which can be mapped to MQSeries message, sent to a Queue to drive an application on any MQSeries platform, and the reply message mapped to other fields in the same Form or in a different one.

**Phase 2:** Non-Notes application can use MQSeries to to populate a Notes database on a Notes Server. (Not to be confused with the Notes Pump function which is intended for scheduled, bulk data movement. MQSeries link is probably better for near-real-time updates involving smaller amounts of data.)

**Phase 3:** LotusScript application on a Notes Client or Server can issue MQI calls via an LSX (LotusScript eXtension).

The phases relate to the delivery sequence of the various functions. You can use each function individually, or in conjunction with others; later phases do not replace earlier ones. Although you could code LotusScript functions that use the Phase 3 LSX to duplicate the Phase 1 and 2 functions, I'm not sure why you would want to?
The reason you might want to use Phase 3 ("MQ LSX") rather than Phase 1 ("MQLINK") is that it is a lot more flexible. The design of MQLINK requires a fixed structure message e.g. Bytes 10-20 of the message map to Notes field XYZ. This may not be ideal if you want to send a Notes text field of highly variable length (and in principle I believe all Notes text fields are of unlimited length). However the main problem with MQLINK in some cases is that the reply message may not conform to a fixed format, especially if you are front-ending an existing application. To a degree MQLINK allows for this, by allowing you to define a string which if present is deemed to prefix an error message (so no attempt is made to update the database). However this may not be flexible enough e.g. If using the IMS bridge, you could get error messages from IMS e.g. If the bridge is not functioning, prefixed "DFS..." and you could get error messages from the IMS application in who knows what format...but not prefixed "DFS...".

So using MQLSX instead of MQLINK gives a lot more flexibility, but using MQLINK is a matter of configuration, whereas using MQLSX is a matter of LotusScript programming - more work and a higher level of skill is required.

The three "Phases" are different and each has it's own strengths. Although there are probably cases where I would still select the MQLink or MQLinkX over the LSX, I can't think of any situation that couldn't be solved with the LSX and a little creativity.

### MQI channel sharing

**Question**

I understand that many clients can share the same MQI channel. Is there any limit on number of clients using the same MQI channel? Is there any implication if a client issue a MQGET wait with long wait time?

**Answer**

No there is no limit. Bear in mind that a Channel definition is just a description of a communication link, not the comms. Link itself. As each client (or message channel) attaches the channel definition is just used as the starting point for negotiation. There is not direct or indirect link to other instances of the same channel definition running.

For example, a server with 10 clients connected each using Channel FRED will still have 10 separate units of execution (eg. threads) running. They will not all share the same thread. As such, there are no consequences from issuing long running APIs from a client such as MQGET (wait).

### MQCONN error AMQ9202

**Question**

I have recently run across the following problem when trying to connect from a windows MQ client application to a QManager on an AIX server: Call to MQCONN fails and the client error file references message AMQ9202. This same client can connect to other AIX QManagers. I have checked the MQ configuration and all seems to be OK, however, I believe the TCP/IP connectivity to be suspect. Has anyone ever encountered this problem?? Any hints would be greatly appreciated.
Answer

The most likely cause for this is incorrect configuration of inetd. Are you sure inetd.conf contains the MQSeries line and the services file has the mqseries port in it. Have you refreshed the inetd subsystem.

The simplest way to test whether inetd is listening is to use telnet. On your AIX box, type telnet AIXHOST 1414 where AIXHOST is the hostname of your AIX machine 1414 is the port you're using.

If you get connection refused then your inetd isn't listening on that port.

MQSeries Win95 Client supports Windows 98

Information

The 'MQSeries for Windows 95 Client' has been tested with MS Windows 98, and supports it.


MQ Client for Java

Question

I am working on a Java-CORBA middle-tier app and hope to use MQ Client for Java in the middle-tier to access OS/390 databases. My question is:

Can I have multiple apps on the middle-tier (AIX) to exchange messages concurrently via MQ Java to a MQ manager on OS/390? In the other words, can I have multiple MQ clients running on the same machine?

Or should I install a MQ manager on the middle-tier and not to use MQ Client for Java. The reason I want to use MQ Client for Java in the middle-tier is that I want to have 100% Java in the middle-tier.

Answer

You can have multiple client instances on one AIX. However your application design should avoid very frequent MQ client instance creation/deletion, as this is a fairly slow & heavyweight process. Creating a client to exchange one request/reply pair is probably not a good idea. Having a Qmgr in the middle tier may offer better performance.

Lotus Notes Remote Agent

Question

In "MQSeries link LotusScript Extension User's manual it mentions Lotus Notes remote agent. What is this?
Answer

Are you referring to the sentence:-

*If you do not have MQSeries installed locally, make use of the remote agent function available within Notes.*

in the section titled 'Designing applications that access non-Notes Applications.'?

What we are trying to say here is that you can use Lotus Notes agents, which can be triggered by an action initiated from a remote Notes client, if you do not have MQSeries installed locally.

The 'MQLSX link sample application', as described in the Release 1.1 MQLSX Users's Guide, uses this mechanism for part of its processing. A document is created on a Notes client and is then copied to an Agent database on the Notes server. The 'creation' of the document in the Agent database causes an agent to be invoked which interacts with MQSeries via the MQLSX.

---

**Question on AccessQueue method of MQLSX**

**Question**

A MQ queue is created as

```vbscript
MQQueueManager.AccessQueue(Name$,Option$,QMName$,DynQName$,AltUid$)
```

If I set model queue name to Name$, and "*" to DynQName$, how can I obtain a dynamic queue name which queue manager assignes.

**Answer**

To get MQSeries to assign a name to the dynamic queue you have to set name¢ to be the name of your model queue. For example:-

```vbscript
Set MQQueue = MQQueueManager.AccessQueue("SYSTEM.DEFAULT.MODEL.QUEUE", Option$, QMName$, "*", AltUid$)
```

MQQueue.name will then a have value assigned by MQSeries.

**Response**

Thanks, now it WORKS!
Data Conversion

Accented chars between OS/2 and AIX

Question

MQSeries for OS/2 v2.0.1

MQSeries for AIX v2.2

I have an MQI setup with a triggering queue on either end (OS/2 and AIX). Since OS/2 does not recognize AIX's 819 codepage, I have set the data conversion to be performed at the AIX side. So, for the Sender channel OS/2-to-AIX, I specify CONVERT=NO, and for the one from AIX to OS/2, CONVERT=YES. I initialize the FORMAT field in the MQMD to MQFMT_STRING (as the data that I pass is chars only) and specify the MQGMO_CONVERT option.

<table>
<thead>
<tr>
<th>OS/2</th>
<th>AIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>sender channel</td>
<td>CONVERT=NO</td>
</tr>
<tr>
<td>FORMAT MQFMT_STRING</td>
<td>FORMAT MQFMT_STRING</td>
</tr>
<tr>
<td></td>
<td>CONVERT=YES</td>
</tr>
<tr>
<td></td>
<td>MQGMO_CONVERT</td>
</tr>
</tbody>
</table>

Yet accented chars are not converted correctly... what may be wrong?

Answer

To help diagnose the problem check the queue manager CCSIDs at each end (use the runmqsc command), the CCSID of the message on the GET and the hex code of an example of what happens: what the character and hex code is at the OS/2 end, what it is on AIX and what you would expect it to be.

I suspect, but may be wrong, that the AIX queue manager CCSID may be 850 not 819 (ISO8859-1).

Data conversion across platforms

Question

If the sender channel has been defined with CONVERT(YES) do you still need to specify MQGMO-CONVERT with the MQGET call? In this case the sender is located on MVS, the receiver on Windows 95. And what to do the other way around?

Answer

Specifying MQGMO_CONVERT on the MQGET call allows the receiving application to specify ANY supported CCSID and encoding as the values to which the message data should be converted.

Specifying CONVERT(YES) on the sender channel allows the message data to be converted ONLY to the CCSID and encoding of the receiving queue manager.
In many cases, the application's CCSID and encoding will be the same as the queue manager's CCSID and encoding, but using MQGMO_CONVERT allows them to differ.

If your application's CCSID and encoding are the same as the queue manager on which it is running, you can use CONVERT(YES) on the channel and not specify MQGMO_CONVERT. However, generally it is recommended to use MQGMO_CONVERT. The reason for this is that if the message passes through several queue managers that have differing CCSIDs and encodings, CONVERT(YES) on each channel may cause the message data to be converted several times, whereas MQGMO_CONVERT will cause the message to be converted no more than once (namely, when the message is retrieved by the receiving application).

### EBCDIC to ASCII Conversion Questions

**Question**

I just started working with MQSeries on AIX. I will be receiving data from an MVS system. The queues are set up with CONVERT=YES (or so I've been told when asked), and the MVS programmer has been able to send (PUT) character data to my local queue. I wanted to use the sample C Get program AMQSGET0.C to get and look at the data in a file.

In the program I noticed that it is setting it's GetMsgOptions with MQGMO_CONVERT. I assumed this would perform any conversion if needed. When I run the program and pipe the output to a file, all the data is unreadable characters.

Can someone explain to me what I'm doing wrong and how to get the data properly? Thanks.

**Answer**

A few suggestions to check:

What FORMAT did the message have when put by the MVS program: If it is MQFMT_NONE (= all blanks) then no conversion will take place. You should get a warning reason code in this case.

Is the sample executable the same date as the source? So early versions of the sample did not have convert set.

Try sending the message to AIX and using the browse sample to check formats, CCSID values and text.

**Resolution**

Thanks for the reply. After sending the message to AIX, I ran the Browse program as you suggested to check formats, CCSID values and text. They are as follows:

```
Encoding: 785
CodedCharSetId : 37
Format : '
```

When I added the GetMsgOpts.Options += MQGMO_CONVERT to the program and recompiled, I did receive the warning reason code (2110 - Message Format Not Valid).
You mentioned that if MQFMT_NONE (= all blanks) then no conversion will take place. Does the program doing the Put have to set the Message Descriptor.Format to MQFMT_STRING (all data is character data), so that the program doing the Get can do the conversion with GetMsgOpts.Options += MQGMO_CONVERT?

**Answer**

To make the messages suitable for conversion you have to change the application on the MVS side. You have to put a move for the MQFMT string.

```
MOVE MQFMT-STRING TO MQMD-FORMAT.
CALL 'MQPUT' USING MQM-HCONN
MQM-HOBJ
MQMD
MQPMO
MQM-BUFFLEN
WK-MESS-TOTAL
MQM-COMPCODE
MQM-REASON.
```

After that, make sure that the attribute for conversion on the sender channel is correct.

Conversion by sender . . . : N Y=Yes,N=No

There is no specific option for the put command. We've been testing this and works fine.

Another method is to use SupportPac MA14, which will do the conversion at the MVS side. We're using this pac for a IBM/MVS - TANDEM environment without any problems.

**Answer**

To confirm and extend what has been said already:

You CAN do conversion in MVS provided you are at version 1.1.4 or later. You must NOT set the Message format to MQFMT_NONE: It must be set to a name which MQ recognizes and can convert. If all your text is characters use MQFMT_STRING. There are other formats which MQ can convert: see the description of the Format field in the MQMD structure described in the Application Programming Reference Manual. If you need to define a new format with character and binary data then you will need to write a conversion exit: see the Application Programming Guide for details.

**Resolution**

To make the messages suitable for conversion you have to change the application on the MVS side. You have to put a move for the MQFMT string.

I followed the above suggestions. After changing the MVS code and doing another Put, the data now comes across converted. Now that I understand what's going on, it all makes sense. Thanks for your help!
**Question**

We are using MQEI and a CICS client to retrieve data from an MVS host. Requests to the host are first sent through a CICS server on AIX. Our queries are working fine, however we are having some data conversion problems with signed decimal data. The data elements in question are defined in the host COBOL program as: PIC S9(09)V9(02). Results are coming back as follows:

- 0000011994C (where C = 3)
- 0000000100{ (where { = 0)

We have tried using the string, long int and short int data types and have changed the character sets and code pages, all with no success.

Should we assume that MQEI cannot properly convert signed, decimal data, and that a conversion should first be done on MVS to make the data into a text string?

**Answer**

Since you are dealing with signed decimal data in your host CICS Cobol application, you would be better off sending it back via the commarea as character data. Applications written in C tend not to recognize signed decimal data.

---

**MVS to OS/2 Data conversion**

**Question**

I'm a little bit confused about that data conversion. I can send some messages from MVS to OS/2 using the mq sample programs, (CSQ4BVK1 on MVS to put messages and AMQSGET on OS/2 to get messages). The messages arrive in MVS, but there is no data conversion. Conversion by sender on the channel definition does not affect the result.

Do I have to use the conversation exit in any case or is there a "simple" method of conversion by the Qmgr? Any hints how to make this conversion running?

**Answer**

If your messages are all text, then to make conversion work between MQM/ESA V113 and MQM/OS2 you need to do the following:

1. Your applications need to set the Format field in the message descriptor to MQFMT_STRING before MQPUT. This is not the default. This must be done both on OS/2 and MVS.

   The most common reason for conversion "not working" is to assume applications have done this when they have not (notably early versions of the IBM-supplied samples).

2. The OS/2 application must specify MQGMO_CONVERT in the Options field of the MQGMO structure before MQGET. This is not the default.

3. The Sender channel from OS/2 to MVS must specify CONVERT(YES). This is not the default.
Response

I checked the options you told me, and, after I changed the sample MVS program (added mqfmt_string to message descriptor) the conversation worked fine - (convert(yes) in channel definition and mqgmo_convert in the sample-get program in os/2 was already defined).

Big Endian / Little Endian

Question

I am looking for guidance/suggestions concerning a "standard" design approach for dealing with binary/hex fields in MQ records which MIGHT cross an INTEL / non-INTEL boundary, such as when the source of the MQ records may be either an INTEL or a non-INTEL platform and the destination is a non-INTEL platform, but might migrate to an INTEL platform sometime in the future.

Answer

From an MQSeries viewpoint, message data can contain integer, character and binary data. Integers are endian, and UNICODE characters are endian (whilst other code pages are not), but binary is not endian.

Each MQSeries message indicates the encoding of the message, including the INTEGER_ENCODING, and an MQSeries receiving program can also determine the encoding of the operating system. Based on this information, a conversion exit can perform conversions on messages as they are read by an application program. There is a supplied default conversion exit for standard MQSeries messages, and this can be overridden by a custom exit for user-defined message formats.

Specifically, the encoding field of a message includes a subfield to define the integer encoding in the message: Two values are defined

MQENC_INTEGER_NORMAL - as on Unix (see note below) and MVS
MQENC_INTEGER_REVERSED - as on OS/2 and NT

There is also an MQENC_NATIVE value defined in the CMQC.H file shipped with each platform.

You could use these values to perform you ouwn byte swap where required or write a conversion exit program to hide this from your application writers.

Note: Some Unix systems run on little-endian processors (including Intel CPUs) which is why it's a bit odd to see references in other MQ books to products being segmented into Intel and Unix arenas.

Also see the description of the "Encoding" field in the MQMD structure in the "Application Programming Reference".

A completely different approach that may or may not suit your needs would be to send all data in character form. If your source data is binary integers, you could use the built-in capabilities of your programming language to convert the numbers to displayeable form (e.g. the "sprintf" function in C). This would avoid the problems of normal and reversed representations of binary integers.
A further benefit is that this data would have a format name of MQFMT_STRING, which is one of the formats that is built-in to MQSeries. That means you don't need to write a data-conversion exit! Hooray!

Only you can decide whether the MIPs consumed in converting between binary and character forms would be acceptable.

Another reason for converting to character data would be if you need to send floating point numbers in your messages. MQSeries does not convert amongst the various floating point representations of the supported platforms, and this is not something I think you would want to implement yourself.

When writing an exit, I suggest one of two approaches:

1. The conversion exit must be able to identify (via header information in the message perhaps) the structure of the message (i.e. 12 characters followed 2 4-byte integers followed by a 2-byte integer, et.) from a set of predefined structures.

2. Create a standard for self-describing messages such that each field of the message is preceded by a description of that content of that field. This description would have to include the type of data (character, short integer, long integer, etc.) and the length of the data if that is not given from the type. The data conversion exit could then automatically convert any message sent in this format. The exit could either retain or strip the description tags during the conversion process.

Option 2 is preferable in my opinion because of its flexibility. In fact, you could use it to enable arbitrary data types beyond character and integer.

---

**CCSID on AIX and HP**

**Question**

We have MQSeries installed on AIX and HP. The Queue-Mgr on AIX has a CCSID of 850, the one on HP 1051. When we start the channels we get an msg that MQS has problems converting the Messages (AMQ9541 "CCSID supplied for data conversion not supported")

- What CCSIDs should be used on AIX and HP (I think 850 is the "multilingual" one and the right one for Europe (exactly Germany) and should be used on both machines).
- How can we change the CCSID of a existing Queuemanager?
- As we can not change the CCSID of the whole Unix System: Can we run the Queuemanager under another CCSID-Environment-Variable than the rest of the system (bearing in mind that we might get errors when an application reads character data from MQS)?
- Can we fix this by using CCSID.TBL?

**Answer**

First: versions 2.2.1 of MQSeries for HP and AIX should support conversion between 850 and 1051. Check you have the latest PTFs on AIX if you are running on AIX V4 as there was a problem in data conversion (but it should not give the symptom you reported).

CCSID 850 is an IBM standard originally defined for use on the PC and is not an international standard.

CCSID 1051, similarly, is an HP specific standard (HP call the codeset name 'roman8').
CCSID 819 is the ISO standard 8859-1.

AIX supports locales with codesets equivalent to CCSIDs 850 and 819

HP supports locales with codesets equivalent to CCSIDs 1051 and 819

MQ supports conversion on HP and AIX between all these CCSIDs.

When you create a queue manager it determines the CCSID from the locale set by the user issuing the crtmqm command.

The default if you do not set the locale (the locale is set to 'C') depends on the level of operating system:

On HP-UX it is 819 and on V10 it is 1051.
On AIX 3.2.5 it is 850 and on V4 it is 819.

To test if the necessary conversion functions of the operating system and the tables provided by MQ are installed you can use the 'iconv' unix command.

On AIX:
```
iconv -f IBM-850 -t IBM-1051 <testfile>
```
will convert <testfile> from 850 to 1051 and display the result.

On HP-UX V9 the command is
```
iconv -f ib850 -t roma8 <testfile>
```
On HP-UX V10 the command is
```
iconv -f cp850 -t roma8 <testfile>
```
The names to use to test CCSID 819 on AIX is 'ISO8859-1' and on HP 'iso81'.

You can not change a queue manager CCSID, it is set when the queue manager is created and depends on the locale, as described above.

On V2.2.1 you should very very rarely have to update the CCSID.TBL file. MQ needs to know some information about a CCSID: for example if its EBCDIC or ASCII, DBCS or SBCS. All the CCSIDs which are in the conversion support tables (they where an appendix in the Distributed Queuing Guide but have been moved to the Application Programming Reference manual) are already coded into MQ. You would only need to add a line in the table if you where adding new conversion tables for a CCSID which is new to MQ.

---

**Question**

We set up two boxes.

1. RS6000 running AIX (English) - CCSID is 819.
2. OS/2 Chinese version - CCSID is 1381.

We install MQ on these two boxes, and build two channels for bi-direction data transfer. But when we start the channel, there is error message:
CCSID supplied for data conversion not support.
and channels can not be startup. The CONVERT option on both channels is set to NO.

Answer

Whether a Channel has CONVERT(YES) or CONVERT(NO) the Channel still needs to do data conversion. Under the covers a Channel must transmit header information (for example the MQMD) which obviously must be converted as we pass from one Queue Manager to another.

Therefore, you can only start a Channel between two machines if at least one of the machines knows how to convert between the two codepages in both directions.

Coded character set conversions

Question

Five questions on character set conversions for string format messages:

1. In which MQSeries manual can I find a description of which character set conversions are supported in MQSeries? Or are there alternate sources of this info i.e. SupportPac? I have found descriptions of how MQSeries allows extensions beyond what it natively supports but I do not know what it supports. I guess it may vary by platform.

2. Are all supported conversions guaranteed to give round trip integrity of data?

3. When 'coded character set' is mentioned instead of 'codepage' does this mean that there are codepoints that the conversion cannot handle so any MQGET can return an error depending on message content?

4. Given that there is somewhere a table of supported conversions where could I get information on what codepoint maps to another codepoint for a given conversion? (This is perhaps an awkward way of asking do MQSeries conversions accord with someone elses standards and if so whose. A manual number would be a nice pointer).

5. Finally, if I have a message containing MBCS data on OS/2 (That is a MQFMT_STRING message consisting of SBCS and DBCS characters) and I pass the message to MVS will MQSeries automatically increase the length of the message when it adds the Shift In / Shift Out (SI/SO) characters?

Answer

1. Where are the conversions listed: In Appendix E 'Code Page Conversion Tables' in the Application Programming Reference (SC33-1673).

2. No, we do not guarantee round trip for characters which are not part of the common characters, partly because the tables used do not all come from a common source.

3. As far as I know the terms character set and codepage are used interchangeably in the documentation.

4. Yes this varies by platform:

   On Current releases here are a few samples:

   **MVS:** MQ code using NLTC tables

   **NT:** MQ code using NLTC tables
OS2: WinCPTranlateString

AIX: AIX iconv function with some mq table additions

HP: HP iconv function with some mq table additions

Solaris: MQ code using NLTC tables for SBCS, Sun iconv for DBCS

AS400: AS400 qtqiconv function.

DEC/OVMS: Dec iconv extended by MQ code to handle EBCDIC DBCS, using NLTC tables.

5. Provided your MQGET buffer is big enough we will extend the message to include the SO/SI characters.

CORRELID and MSGID

Question

Using both MQSeries on AIX and MVS.

On both side, applications set CorrelID in the message descriptor to specify necessary messages.

Sending applications set string data to CorrelID. But receiving applications does not consider it as string.

I understand CorrelID and MsgID are MQBYTE24 type data. Does it mean that these IDs are "byte stream" (not string)? MQSeries do not try to convert string IDs, do they?

Let me know, how I can use the common IDs between AIX and MVS.

Answer

The CorrelId field of the message descriptor is treated as a string of bits and not as a string of characters and hence is not subject to any translation (this is described in the MQSeries Application programming Reference, and in the MQSeries Technical Reference, in the section which describes the fields of the MQMD).

To convert the CorrelId, you could:

1. Let the server applications at both ends convert the CorrelId field immediately after getting messages from the request queues, but before actually processing the messages.

2. Let the server and the client application at one end only, convert the CorrelId field immediately before getting a message from the local request queue, or putting a message to the remote request queue.

3. Write Message exits on the sender channels at both ends, or on the receiver channels at both ends, or on the sender and the receiver channel at one end, to convert the CorrelId field. The Message exit is passed the transmission queue header (MQXQH) and the application message text. The MQXQH contains the Message Descriptor (MQMD), which in turn contains the CorrelId you want to convert.

Note: The format of the MQXQH can be found in the MQSeries Application Programming Reference manual, and further details of Channel exits can be found in Chapter 6 - User Exits, of the MQSeries Distributed Queue Management Guide.

SupportPacs MA11 (MQSeries for MVS/ESA - ASCII to EBCDIC Conversion Sample) and MA14 (MQSeries for MVS/ESA - ASCII to EBCDIC Conversion Sample) contain sample message exits for the CICS mover and
the non-CICS mover respectively. However, they only deal with conversion of the user text, and not with conversion of the Message Descriptor. Nevertheless, they should give you a good starting point.

More Details

The "expected" use for the "MsgId" and "CorrelId" fields is described below. In these examples, "client" means the application sending a request message, and "server" means the application sending the reply message.

1. The client sets: MsgId=MQMI_NONE and CorrelId=MQCI_NONE and sends the message. The client does not therefore need to worry about ASCII or EBCDIC at this point.

2. The client's queue manager changes the "MsgId" in the request message to a unique value, and returns this value in the client's MQMD on output from MQPUT.

3. The client saves the value returned in "MsgId" on output from MQPUT in a variable the client declares called (for example) "SavedMsgId".

4. The server gets the message. In the reply, the server sets CorrelId=received MsgId, and MsgId=MQMI_NONE. The server does not therefore need to worry about ASCII or EBCDIC.

5. The server's queue manager changes the "MsgId" in the reply message to a unique value.

6. If the client sends only one request at a time, it will be expecting only one reply. It therefore issues MQGET with the wait option and with MsgId=MQMI_NONE and CorrelId=MQCI_NONE. The client does not therefore need to worry about ASCII or EBCDIC.

7. If the client sends many requests at a time, it will be expecting many replies:
   a. If the client wants to wait for a SPECIFIC reply, it issues MQGET with the wait option and MsgId=MQMI_NONE and CorrelId=SavedMsgId. The client does not therefore need to worry about ASCII or EBCDIC at this point.
   b. If the client wants to wait for ANY of the expected replies, it issues MQGET with the wait option and MsgId=MQMI_NONE and CorrelId=MQCI_NONE. To check WHICH reply it has got, it compares the value returned for "CorrelId" with the values for "MsgId" that it saved when the messages were put; the one that matches correlates the reply with the request. The client does not therefore need to worry about ASCII or EBCDIC at this point.

The overall result with this style of use is that neither the sender (client) nor receiver (server) need to worry about ASCII or EBCDIC as far as the "MsgId" and "CorrelId" fields are concerned.

However, the expected use of the "MsgId" and "CorrelId" fields is not enforced by the queue manager, so applications can use them for whatever purposes they choose. In particular, the sending application could put a readable string in the "CorrelId" field, and the receiving application could try to get an expected message by issuing the MQGET call with CorrelId='MY_READABLE_STRING'.

If the applications choose to use the "MsgId" and "CorrelId" fields in this way, the applications must take the responsibility for translating between ASCII and EBCDIC, in those situations where it is needed (e.g. AIX to MVS). Also bear in mind that in multi-platform environments, a receiver might get some messages with ASCII ids, and others with EBCDIC ids, so it would need to adopt some strategy for deciding whether it needed to translate the ids.

For example, one solution would be always to set the first byte of the id to a specific character; then, if that byte in the received id matches the expected character, the id is in the receiving application's character set; otherwise it is in the "other" character set. For this to work, you would need to restrict your character ids to characters that are invariant in all of the various CECP character sets; I would recommend using only upper and lower case letters, plus numeric digits. Don't use letters with peculiar accents, or special characters such as punctuation.
Considerations

However, this can lead to another set of problems, so read on.

If the messages on a queue might have some ids in ASCII, and others in EBCDIC, the receiving application cannot issue an MQGET with a character "MsgId" or "CorrelId", UNLESS it has already determined the character set being used for the ids in the message required, and set "MsgId" and/or "CorrelId" on the MQGET to the character values required in the appropriate character set (ASCII or EBCDIC). It could make this determination via a browse MQGET that specified MQMI_NONE and MQCI_NONE, but in truth this is probably a bad idea because of the increase in path length.

Character ids are feasible ONLY if the receiving application does not need to use them until AFTER it has retrieved the message from the queue (i.e. the application always uses MQMI_NONE and MQCI_NONE on its MQGET calls).

Additional thoughts on data types

The MQI is defined in terms of "abstract" data types, such as MQLONG and MQCHAR48. Although you might think that abstract data types are not very useful, these abstract data types actually map very directly to user-defined data types in C.

The reason for using abstract data types is to try to make a distinction between "things" that are logically different. So, for example, we have MQLONG for numbers, MQHCONN for connection handles, and MQHOBJ for object handles, even though all of these actually map to the same eventual data type (a 4 byte signed binary integer on the systems currently-supported).

If you look at the string data types, you will notice that some are declared as "MQCHARn" (e.g. MQCHAR48), and some as "MQBYTEn" (e.g. MQBYTE24). The distinction that we were trying to make here is with respect to character-set translation (e.g. ASCII<->EBCDIC):

- An "MQCHARn" data type is a string of PRINTABLE CHARACTERS, and it therefore needs translation when moved between systems with different character sets. Fields with "MQCHARn" data types in the message descriptor are converted by the message channel agent (MCA), so the application sending or receiving a message does not need to convert them (unless you are writing an MCA!). The "ReplyToQ" field in MQMD is like this.

- An "MQBYTEn" data type is a string of UNPRINTABLE BYTES (i.e. bits), which we interpret to mean a string which must not be translated when moved between systems with different character sets (because the bytes do not represent printable characters). The "MsgId", "CorrelId", and "AccountingToken" fields in MQMD are like this.

As far as the message descriptor is concerned, the ONLY fields which will not be converted by the MCA are those with "MQBYTEn" data types; all other fields will be converted as necessary by the MCA.

Data Conversion OS/2 - MVS

Can someone help me to clarify what needs to be done to get application data conversion when sending character data from Windows clients via MQSeries for OS/2 to MQSeries for MVS and back again.

All messages will go to/from IMS applications in the MVS-system. The TCP/IP mover is being used.
If I understand the DQM correctly it would be possible to let the distributed system do all the conversion by specifying:

- For the sender channel to MVS: CONVERT attribute
- For messages coming from MVS: Conversion at MQGET time.

1. Is this correct, or do we need to have a message exit on the MVS system?
2. What other specifications are needed? CCSID, ...?
3. Will the conversion work all they way from/to the Windows clients?
4. The customer will later connect also MQ on OS/400 and on DEC VMS to his MVS system.
   a. For OS/400 I suppose the conversion in both directions could be done by MQ OS/400. Correct?
   b. For DEC VMS I suppose we need a message exit in MVS. Is there a sample available for the TCP/IP mover?

All hints would be very much appreciated.

Answer

1. Yes this is correct. You do not need a message exit on the MVS system.
2. The MQGET needs to specify the CCSID in the message descriptor as being that of the client.
3. Yes
4.
   a. Yes MQ/400 also supports conversion in both directions
   b. I expect that you need an MVS message exit if you're going to DEC VMS. Try looking at SupportPac MA14

Questions Ref: Clients & Sender channels..

Question

In the Distributed Queueing Guide, the statement is made that data conversion should be done at the receiving Qmgr (channel). Yet I have seen the OS/2 manuals say that CONVERT should be set to CONVERT(YES) for sending channels. I believe that now MQM/ESA V114 does do user data conversion, the data conversion should be done automatically at the receiving channel (on an OS/2 to MVS Sender) not at the OS/2 sender. Is there some reason to do conversion on the (OS/2) sending side?

Answer

If you are going to do data conversion on a channel, it can only be done at a sender or server channel and is specified by the CONVERT(YES) channel attribute. Conversion at the receiving end would be done in your application by specifying MQGMO_CONVERT when getting your messages from the destination queue. In general, this latter technique is recommended for two reasons:

1. so that in multi-hop channel scenarios where the Qmgrs are running on different codepages, conversion only need be done once at the end, not at each intermediate Qmgr
2. when the channels are busy, data conversion is an unnecessary overhead which is better done when the application needs the message (which may be some time later). For example, some messages may expire before they are got by an application, so it would be wasteful to convert these on a channel if they aren't going to be needed.

MQSeries for MVS/ESA v114 does support user data conversion and so there is no need to do the conversion at your OS/2 sender end.
General

Authority for RUNMQSC commands

Question

I had administrator in the MQM group but even though the id could start a Qmgr, when the id issued runmqsc it got AMQ9508 'cannot connect...". Seems strange, but I looked around in the MQ NT book and found on page 93 "For MQ authorizations... Are limited to a maximum of 12 characters." So is that it, because administrator has 13 characters in it the id cannot be authorized to MQ?

Answer

Yes. MQ ids can only be 12 characters long, and as you have noted, 'administrator' has 13.

Basic MQSeries concepts

Question

I'm trying to get a handle on how all the pieces of MQSeries (queues, queue managers, and channels) fit together. I've read a bit in the AIX System Management Guide and the AIX Distributed Queueing Guide and I'm a little confused. I'd appreciate answers to the following questions or a pointer to a "tutorial-level" explanation of the overall MQSeries architecture.

1. A channel connects two "adjacent" nodes, which implies a channel cannot be used to connect nodes that are not "adjacent". How does MQSeries determine whether two nodes are "adjacent" or not?

2. True or false: a channel connects exactly one queue to exactly one other queue. (And one can get the effect of multiple channels to a single queue by using aliases.)

Answer

For this purpose, nodes are "adjacent" if you can have an LU6.2 session between them, or a TCP/IP sockets connection. In other words, if the communication protocol you are using can talk between those two nodes.

You usually have one channel pair between two queue managers. You can write to many remote queues over one channel.

Basic questions
**Question**

Suppose I have a link between two nodes, with messages enqueued on the transmission queue, and Something Happens to make the link Go Away.

1. Is it true that when the link Comes Back the enqueued messages will automatically be sent?
2. Is there some way a third party (e.g., an application) can discover that the link has Gone Away?
3. Is there some way a third party can redirect the messages on the transmission queue to a different link (e.g., if the link is not going to Come Back for some time)?

**Answer**

1. If the link goes away the channel will go into retry (assuming retry counts are non-zero) and should restart when the link comes back.
2. An application can look out for the event messages which are generated when a channel stops for whatever reason.
3. An application could redirect the messages but you must take account of any messages which are indoubt when the link fails.

---

**Channel Connection Name**

**Question**

In MVS/ESA MQSeries using distributed queuing with CICS, I have used the CICS panels to create a channel, but I did not know what to put for the CONNAME field. I am temporarily using "ABCD" for that, but will need to know what should really be there when we start up the system. I suspect some sort of connection type or ID needs to be specified there, but I know very little about CMIS, and don't know where to look for it.

My system support person is out of town for a while, and we need to get going soon.

Also, are there any specific names or relationships that need to be used for local, remote, and transmission queues; channels; and message queue agents, other than that the same name must be used at each end of a channel?

**Answer**

A connection name of ABCD is fine as long as you have a CONNection defined in your CICS system of that name. If you have not then you can use CEDA DEF CONN to provide one. You need to provide the LU name in the definition of the target system.

You will also need SESSion identifiers associated with your connection ABCD. Again CEDA can be used to define sessions. (CEDA DEF SESS) Here you should use protocol APPC.

There are no special rules for names of the various objects that you mention ONLY that the CHANNEL NAME must be the same at both ends.
Question

Perhaps I should back up a step or two before asking about making CICS connections. All I see in the manual for MVS/ESA using CICS is LU 6.2 connections, but the connection we need is from MVS/ESA to the AIX environment, which uses TCP/IP. Is there even any way to involve CICS for such a channel?

If so, what needs to be done about an LU name for the target system? If not, do we just ignore CICS for that part of MQSeries and pretend we are not using MVS/ESA with CICS.

As I said before, I know very little about CICS. If it is needed, we will have to find someone who can check out what has already been done in the CICS environment and locate whatever names, etc. That are already defined. Or is there a transaction that does that? With that information, defining the CONN and SESS should be doable.

Answer

You are correct. The CICS mover ONLY supports LU6.2 channels. So unless you put SNA on your AIX box you will need to use TCP, i.e. A non CICS mover.

Also, you should be able to use the CICS mover for LU6.2 connections and TCP- non CICS for TCP connections from the same MQM. BUT, I would certainly recommend avoiding use of the CICS mover and so for LU6.2 use APPC/MVS instead.

Remember that the MQSeries for MVS/ESA's DQM feature using CICS does NOT support TCP/IP.

You will have to use the native DQM feature. This would always be our recommendation.

Deleting expired msgs from xmitq

Question

How can I avoid MQRC_Q_FULL messages when there are only expired messages on the queue?

It seems as though the only way is to do a browse while there is still an unexpired message on the queue.

Is this the only way?

Answer

You may be right, although I'm not certain that there needs to be a message there. There is a program on
http://www.software.ibm.com/ts/mqseries/txppacs/mo01.html

which you can run regularly which will browse all your queues in order to throw away expired messages.
Events Reading

Question

Has anyone any Cobol samples where you read eg.SYSTEM.ADMIN.CHANNEL.EVENT or other of these Admin queues?

Answer

Please see SupportPac MS12 which includes a COBOL program, READPCF, which receives and formats event messages. It was written for the MVS platform and formats many channel events. For events it does not recognize it formats the PCF structures.

It is designed to run as a batch program and to be passed the queue manager and queue names.

Message Priority

Question

My customer has several applications which are PUTting messages to remote queues and he wants some of them to be delivered first and some of them to wait if important ones are coming.

QUESTION: Does a messages with priority 3 for example have to wait in the transmission process (maybe in the XMIT queue) if there is also a message with priority 1? Is using message priority a way to help his important messages be delivered more quickly over the network than less important ones. We know there is the possibility of having separate channels and XMIT queue but then he has a lot more definitions (he has about 150 servers).

Answer

Yes, you can certainly use message priority to send a message to the front of the queue. But the priorities are the opposite direction. Priorities go from 0-9 (version 2 Qmgrs only of course) with 0 being the lowest and 9 being the highest. (A priority 9 message will get processed before a priority 1.)

Question

Does MQ have to read the whole queue-chain, if I want to work with priority or are there 9 priority-chains in one queue? What is the effect on performance?
If the queue is defined with MSGDLVSQ(PRIORITY) then there are essentially 10 internal queues, one each for priority 0-9. So I don't believe there is any performance hit in using priorities.

**MQ in comparison to RPC and CICS**

**Question**

I attended one of the MQ sessions at SHARE. I will admit some confusion concerning how MQ, RPC and CICS should be used. When would we use each of these tools?

By "CICS", I am referring to the concept that one could install CICS on many platforms (MVS, OS/2, HP-UX, etc.) and use CICS as "middleware" so that applications on the various platforms could communicate together. (That is, perhaps CICS is NOT just for "transaction processing".)

Now, to some degree, that paragraph sounds just like "MQ" to me. Also, from what little I know about OSF's DCE RPC (Remote Procedure Call), it also might fit that mold.

Perhaps I am on the wrong track, but I have the feeling that we should choose one of those "plans" as the standard connectivity tool for our systems.

Any clarification you can give on that concern would be appreciated.

**Answer**

Perhaps I can help you with your question. You wonder about CICS, RPC, MQSeries, and even DCE. Let's start with the basics.

CICS is a different animal from MQSeries, RPC or DCE. CICS is an Application Server (or OLTP or Transaction Monitor) which means it is a manager of resource managers. It can handle multiple different resource managers a control update integrity across them all. It has an application based end to end view.

MQSeries is a resource manager, managing the resource of communications. RPC is somewhat similar although it is really for internal use in Encina. DCE is many things!

So what do you use for interoperating between systems (at least that is what I think your question is)? What do you use for distributed apps?

The answer is it depends on what you want. A fundamental difference between DCE and RPCs (Remote Procedure calls), and CICS on the one hand and MQSeries on the other is that CICS and RPCs are basically synchronous. Hence, when you ask for a service from a server system both data AND CONTROL are transferred. Your requesting application waits for the procedure to run and for the answer WITH CONTROL to come back.

This is an intuitively simple model. Excellent for providing realtime synchronicity between systems because of the serial nature.

MQSeries is basically asynchronous. When a service is requested from a server the information moves BUT CONTROL DOES NOT. Hence the requesting application can continue doing other things. Because of this
MQSeries has a natural way of coping with the lack of a connection. If a connection is unavailable, MQSeries just queues the request up and it is executed once the system is available. Or if a system is busy the request will be held until the system is ready.

The MQSeries model allows business processes to be decomposed and executed in parallel since there is no longer a serial synch requirement. However data is 'near to real time' synchronous rather than real time. In most applications this is satisfactory.

But, to reiterate, if you want coordination across updates made by multiple different resource managers then you better have CICS. MQSeries can’t help.

**MQS administration interface**

**Question**

Is there a tool which provides a text or PM interface to MQS OS/2? We would like to visualize the channels, status, curdepth, and execute commands on MQ remote channels. I know there is an API to do that, but has somebody developed an interface?

**Answer**

What you need is the MO31 MQSeries for OS/2 Remote Queue Administrator Supportpac which gives you a PM interface. It allows you to define, operate and monitor MQ resources in local and remote queue managers. The latest update allows you to manage MVS systems too. SupportPacs are available from the MQSeries web page.

A text based interface is provided by the runmqsc utility.

**Response**

Thank you for your response, the MQSeries Monitor Program is great. We can manage local and remote queues through a more friendly interface than RUNMQSC.

**MQSeries List Server**

**Question**

Is there an MQseries listserver I can subscribe to?
Answer

A listserver facility has been established for MQSeries. It is hosted by The University of Vienna. This facility means that, once subscribed, a user can E-Mail to the listserver and the note will be distributed to all other subscribers. It thus allows MQSeries users to have discussions amongst themselves without relying on IBM hosted forums.

To subscribe send a note to listserv@akh-wien.ac.at with a line saying sub mqseries Firstname Lastname. You will then be sent a confirmation note which you will need to reply to.

Once subscription is complete you can send queries via EMail to the mailing list address which is MQSeries@akh-wien.ac.at (or MQSER-L@akh-wien.ac.at).

MQSERIES PROCESS NAME & DESCRIPTION

Information

The purpose of this item is to document the Process Names and their descriptions that are used by MQSeries on Specific platforms.

Platform

<table>
<thead>
<tr>
<th>Process</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX</td>
<td></td>
</tr>
<tr>
<td>amqhasmx</td>
<td>logger</td>
</tr>
<tr>
<td>amqharmx</td>
<td>log formatter, used only if the queue manager has linear logging selected</td>
</tr>
<tr>
<td>amqzl1p0</td>
<td>checkpoint processor</td>
</tr>
<tr>
<td>amqzlaa0</td>
<td>queue manager agent(s)</td>
</tr>
<tr>
<td>amqzxma0</td>
<td>processing controller</td>
</tr>
<tr>
<td>amqcrsta</td>
<td>TCP/IP Receiver channel &amp; Client Connection</td>
</tr>
<tr>
<td>amqcrs6a</td>
<td>LU62 Receiver channel &amp; Client Connection</td>
</tr>
<tr>
<td>runmqchl</td>
<td>Sender Channel</td>
</tr>
<tr>
<td>runmqsc</td>
<td>MQ Command interface</td>
</tr>
<tr>
<td>amqpcsea</td>
<td>PCF command processor</td>
</tr>
<tr>
<td>AS/400</td>
<td></td>
</tr>
<tr>
<td>AMQHIXK4</td>
<td>Storage Manager (Housekeeper)</td>
</tr>
<tr>
<td>AMQMCPRA</td>
<td>Data Store (Object Cache)</td>
</tr>
<tr>
<td>AMQCLMAA</td>
<td>Listener</td>
</tr>
<tr>
<td>AMQALMP4</td>
<td>Check Point Process</td>
</tr>
<tr>
<td>AMQRMCLA</td>
<td>Sender channel</td>
</tr>
<tr>
<td>AMQCRSTA</td>
<td>TCP/IP Receiver channel &amp; Client Connection</td>
</tr>
<tr>
<td>AMQCRS6A</td>
<td>LU62 Receiver channel &amp; Client Connection</td>
</tr>
<tr>
<td>AMOPCSVA</td>
<td>PCF command processor</td>
</tr>
<tr>
<td>AMQRIMNA</td>
<td>Channel initiator (trigger monitor to start channel)</td>
</tr>
<tr>
<td>AMQIQES4</td>
<td>Quiesce (forces user logoffs - for upgrades)</td>
</tr>
<tr>
<td>AMQIQEJ4</td>
<td>Quiesce (without user logoffs - for daily use if desired)</td>
</tr>
</tbody>
</table>

ATT/GIS
amqhasmx - logger
amqharmx - log formatter, used only if the queue manager has
    linear logging selected
amqzllp0 - checkpoint processor
amqzlaa0 - queue manager agent(s)
amqzxma0 - processing controller
amqcrsta - TCP/IP Receiver channel & Client Connection
amqcrs6a - LU62 Receiver channel & Client Connection
runmqchl - Sender Channel
runmqsc - MQ Command interface
amqpcsea - PCF command processor

HP/UX

amqhasmx - logger
amqharmx - log formatter, used only if the queue manager has
    linear logging selected
amqzllp0 - checkpoint processor
amqzlaa0 - queue manager agents
amqzxma0 - processing controller
amqcrsta - TCP/IP Receiver channel & Client Connection
amqcrs6a - LU62 Receiver channel & Client Connection
runmqchl - Sender Channel
runmqsc - MQ Command interface
amqpcsea - PCF command processor

OS/2

AMQHASM2.EXE - The logger
AMQHARM2.EXE - Log formatter (LINEAR logs only)
AMQZLLP0.EXE - Checkpoint process
AMQZLAA0.EXE - LQM agents
AMQZXMA0.EXE - Execution controller
AMQXSSV2.EXE - Shared memory servers
AMQCRSTA.EXE - TCP/IP Receiver channel & Client Connection
AMQCRS6A.EXE - LU62 Receiver channel & Client Connection
RUNMQCHL.EXE - Sender Channel
RUNMQSC.EXE - MQSeries Command processor
AMQPCSEA.EXE - PCF command processor

SINIX

amqhasmx - logger
amqharmx - log formatter, used only if the queue manager has
    linear logging selected
amqzllp0 - checkpoint processor
amqzlaa0 - queue manager agent(s)
amqzxma0 - processing controller
amqcrsta - TCP/IP Receiver channel & Client Connection
amqcrs6a - LU62 Receiver channel & Client Connection
runmqchl - Sender Channel
runmqsc - MQ Command interface
amqpcsea - PCF command processor

SUN/OS
Question about when Messages Expire

**Question**

If we set a message to expire, when does the Queue Manager actually get rid of it.

Is there any way in which to configure the Queue Manager so that it will go ahead and delete these expired message the first time it comes across them (while looking for other messages?)

I want to make sure that I don’t bother getting any messages which are no longer useful because the information in them is already old.
Answer

Please read the Application Programming Reference, Chapter 1, where the Expiry field of the MQMD structure is explained. Here you will find that it is not defined precisely when a message that has expired is actually physically discarded. However, you can be sure that a message that has expired will never be returned to an application - the act of trying to get the message will cause it to be physically discarded.

Response

I've seen queues reject valid messages because it was filled with expired messages. Admittedly, this occurred under stress test situations, but I found no reason to believe that it couldn't happen in production.

Is there a way to 'force' the queue to dump expired messages?

Answer

Ideally, there ought to be a "daemon" QM process that spends its time looking at queues and discarding expired messages. But there isn't. With the current implementation, expired messages are discarded only when an MQGET is performed THAT WOULD HAVE RETURNED THE MESSAGE HAD IT NOT ALREADY EXPIRED.

Triggering could be used to start additional instances of the server application when the queue depth exceeds a specified threshold (or start the first instance, if the server is not yet running), and event messages could be used to alert a system monitoring program or the administrator.

Use of datagrams and replytoQ

Question

I have read that datagram message types should be used when you do not require a reply from an application. I have a couple of questions regarding this. What is the rational for this restriction? Is it performance related? What type of message should be used if a reply is not necessarily required? I am using datagrams in a application that reads the replytoQ name from incoming datagram messages. Depending on application specific scenarios an outbound message or messages may be sent. The outbound message is a datagram messagetype and is sent to the queue specified in the incoming messages replytoQ name. What are the problems using MQ in this manner?

Answer

Using MQMT_REQUEST for messages that expect replies, and MQMT_DATAGRAM for ones that do not, is the suggested way of using these values, but this is not mandated by MQSeries. So there is no problem with using MQMT_DATAGRAM for messages that sometimes have replies and sometimes do not.

Note the following points:

1. If you specify MQMT_REQUEST, the queue manager will require you to specify a non-blank "ReplyToQ" in MQMD.
2. If you specify MQMT_DATAGRAM, "ReplyToQ" can be blank or non-blank, but whatever value it contains is transmitted from the sending application to the receiving application.

3. Some server applications may REQUIRE you to specify a particular value for "MsgType", that is, the server may reject your message if you specify the wrong value. Whether it does this depends on the designer of the server application. If you are designing both requester and server parts of the application, you have both parts under your control, and can design the application as you wish.

Using MQI to monitor and controlling channels

Question

I have done a fair amount of programming MQI for access to MQ queues in C. It would be useful to have the ability to programmatically monitor and control MQ channels. The System Management Guide (NT V2.0) indicates on page 43 that this should be possible. The application Programming Guide (in describing MQOPEN) also indicates how one can open a channel object. However, from this point on I have not be able see how to proceed. It is not clear what programming constructs can be used to inquire about channel attributes or status and how to start or stop a channel.

Can somebody point me to some documentation or some sample programs that interact with MQ channels?

Answer

The programming interface you want is PCF. Take a look at the MQSeries Programmable System Management book.

Note that you cannot MQOPEN a channel object. You can only OPEN queues, processes, queue mgs and namelists (MVS).

To inquire / modify / start / stop channels, you must use the MQ command queue and send a PCF formatted message to it. See MQSeries Programmable System Management SC33-1482-05, for more details.

When do you really need MQ Series "Client" component vs server?

Question

When does an application really need to have the MQ Client component as part of its configuration? vs. the alternative of accessing the client's queue on the MQ server directly from the client application. My understanding is that "MQ Client" mainly allows you to maintain your own queue at the client workstation. Is this correct?
Answer

The "MQ Client" is NOT there to maintain queues on the local (client) workstation. In fact, exactly the opposite is true. Use the MQ client when you want to connect "remote" queues, queues that are sitting on a server. The product that you might be thinking of is the lightweight or "leaf" queue manager for windows. That is a "small" server that allows you to keep queues on the client PC.

A good book to read is the "MQSeries Clients" book that come with the package. Chapter 1: What is an MQI Client and Why use MQI clients.

XMIT Q truncation feature

Question

Regarding queue size and xmit queue characteristics:

1. How many 4meg messages can fit into one queue?

   Xmit Queue access

   Background scenario:

   I placed a 1-megabyte message on our local test queue and retrieved it successfully. I placed the identical message on our XMIT Q. But when I attempted to retrieve it locally, I get the following return code:

   W03-COMPCODE 00000001
   W03-REASON 000002079,

   Which is caused by the return of a truncated message. The data in the flat file I created post-retrieval from the XMIT queue is offset dramatically.

   The same COBOL programs and JCL were used to PUT and GET both messages. The GET program has the MQGMO-ACCEPT-TRUNCATED-MSG feature associated with the MQGET call. I accessed the queue just to verify what had happened.

2. What would cause the GET from the XMIT queue to ignore the truncation feature?

3. Are XMIT queues significantly different to work with?

4. Currently our channel with the Unix machine is not in place. Would this possibly be cleared up somehow after transmission has taken place?

Answer

The behavior you are seeing is exactly what is expected.

When a message is put on a transmission queue, some header information is added to the message (about 500 bytes). This header contains the information needed by the remote queue manager to determine what to do with the message when it arrives (such as the destination queue name). The header is the MQXQH, described in the Application Programming Reference.
Usually you would not read message directly from the xmit queue. The channel program would usually read the message and send it across the link where the remote channel program will process the header. When the message reaches its destination the header is removed.

The 2079 reason code is not ignoring the ACCEPT_TRUNCATED_MESSAGE. Even when you use this option you get a warning when truncation occurs but the message is retrieved from the queue.

If you do not use this option you get reason code 2080 and the message is not removed from the queue.

I imagine that your application is expecting 1 Meg of data but because of the header it is trying to get a message which is 1Meg + 500 bytes.

**EDI and MQSeries**

**Information**

EDI (Electronic Data Interchange) is a broad term which refers to paperless communication of business documents (purchase orders, bills, payments, etc) usually thru a VAN (Value-Added Network) supplier (Advantis, GE/IS, etc) which provides guaranteed delivery and associated services like archiving and backup and recovery. It is asynchronous. The documents conform to international standards as to definitions of records and fields: order, width, etc.

The records all have specific identifiers as to type and are contained in a document "envelope" with control totals, so lost data is detectable. The VAN's handle multiple companies, any two of which that communicate are called "business partners"..... A supplier of manufacturing tools might have many customers. GM demands that all its suppliers conform to their variation of the document standard, for example.

(VANs and other software vendors provide software packages to connect to their network, including formatters (Translators) to take a legacy system file and rearrange data to the standard documents and fields.)

The business partners do not have a connection to one another, but to the VAN service. Most often, documents can be put out to the VAN in batches (when a legacy system runs at night). The business partner "pulls" the messages from the VAN at its convenience, later. As input to another legacy system.

MQSeries is software to handle asynchronous communication of data, but can be used for synchronous communication also. It is assumed that the "business partners" in this case are constantly connected. MQSeries advantage is that it can run/connect different platforms (MVS, AIX, OS/2).

"**MQ Directory" - MQSeries resources on the WWW**

**Question**

See http://www.software.ibm.com/ts/mqseries/directory

MQ Directory can help you easily find MQSeries-related products and services on the World-Wide Web. It includes listings
If you know of products or services that should be included, please let us know. Use the 'feedback' button on the Directory pages.

Are transmission queues safe during a failure

Question

If a remote Qmgr is down and messages are queuing for it in a local transmission queue, is this local transmission queue on disk and therefore safe from a machine crash or will all be lost if the local machine crashes?

Answer

This depends on your message attributes. With persistent, then your messages survive a restart of the machine, when not all messages are lost.

Persistence is an attribute of each MESSAGE, and is established by the application that does MQPUT when it creates the message. A message is either persistent (and survives queue manager restarts) or is non-persistent (and erased at queue manager restarts). Any given queue, including transmission queues, may contain a mixture of persistent and non-persistent messages.

The MQPUT verb permits you to specify three possibilities for persistence

1. The application defines the message as non-persistent.

2. The application defines the message as persistent.

3. The application would like the message to be persistent or non-persistent based on some default attribute, which is specified with the queue. This is where the "default persistence" queue attribute comes in. If the application says "use the queue default", then the message inherits its persistence attribute from the queue definition. However, if the application says "persistent" or "non-persistent", then the queue-defined default persistence attribute is not used. (see Command Ref, define ql keyword=defpsist)
Question

For a given set of messages, I want to have the following:

- multiple applications adding messages to the set
- multiple pools of applications, each pool pulling messages from the set
- within each pool, a single application processing each individual message

I put the statement of the problem in terms of a "set" of messages rather than a queue because there may be multiple queues involved.

I'm not sure that what I have above is understandable, so let's explore a really bizarre example which will hopefully explain what I want to do.

Let's say we're studying rabbits living in some valley. We have observers A and B wandering through the valley observing the rabbits, which have identification tags making it easy for the observers to see which rabbit is which. A and B send messages about the rabbits to a central point, and teams of researchers process those messages. Team 1 (made up of scientists T and U) is interested in building a comprehensive family tree. Team 2 (made up of scientists V, W and X) is studying the diet of the rabbits. Team 3 (made up now of only scientist Y, but scientist Z may join her off and on) is studying the health of the rabbits, including birth, death, predators, diseases, etc. Each team has one or more scientists which are collaborating on their particular problem. Individual messages from A and B might include statements like "Rabbit J1 ate 3 leaves from a squash plant, which made him sick for an hour," or "Rabbit K3 had 4 babies this morning, designated M1, M2, M3 and M4, and they all nursed for 15 minutes before falling asleep." Each message, therefore, is potentially of interest to each team, thus each team needs to get a copy of the message to analyze. (Oh, and each team must receive only a single copy of each message.) Within a team, however, only one scientist needs to process a message, since the scientists collaborate so well with members of their own team. Since each scientist works at their own pace, though, we don't want a simple round-robin distribution scheme within the teams, we want a single list of messages the members of a team can grab from when they're ready for more work.

Now if we want to implement this messaging scheme with:

- applications for observers A and B
- applications for each scientist
- MQSeries as the mechanism for getting messages back and forth

Then what queue architecture do I need to implement? Do I need applications for each team to pull messages from one queue and put them on another?

How does the architecture change if I want it to be general enough to support the following?

- a variable (and unknown when the code is being written) number of observers

- a variable (and unknown when the code is being written) number of teams

- a variable (and unknown when the code is being written) number of scientists within each team

What mechanisms can be used for cleaning up messages if teams may be added or removed at will?

By the way, I don't have much MQ background, so the more administrative and programming details you can provide, the better! I realize this is probably a large question, so I truly appreciate any help provided.
Answer

There are obviously many different ways of approaching this, but here's a scheme that has MQSeries do most of the hard work. The problem is really split into two parts: the distribution of the message to multiple interested groups, and the management of contention among individuals in a group for each message.

Here's the scheme that leaps to mind:

- destination queues are assigned: one queue per pool of applications
- the collection of destination queues is used to form a distribution list
- the sender applications (observers) send messages to the distribution list -- this results in a separate copy of each message arriving on the queue for each pool (team of scientists) without the observer having to generate multiple copies of the message
- any number of the applications in a pool can be running against their pool queue concurrently, simply sitting in a loop reading the next available message. Once one application has retrieved a particular message, it is no longer visible to the other applications in the pool, even though they are looking at the same queue. Other pools have their own copy of the message on their own queue to deal with at their leisure

On scaling, with this scenario:

- there can be (to all intents and purposes) any number of sender applications (observers), subject to the throughput of the pool applications (scientists)
- the number of application pools (teams of scientists) is limited by the number of entries you can fit in a distribution list (without really thinking about it or trying it, hundreds at least)
- the maximum number of applications in a pool (assuming they might all be active at once) is determined by the maximum number of concurrent handles allowed on the destination queue/queue manager (hundreds, generally)

To add a new team:

- define a team/pool queue, and add it to the distribution list used by the observers

To remove a team:

- remove the queue name from the observers' distribution list
- delete the team queue (other teams won't know or care)

To add a new team member (app in a pool):

- fire up a new instance of the team app against the team's (existing) queue

Other things:

- to use the distribution facility of MQSeries, the observer apps must be running MQSeries V5 -- but destination queues/pool apps could be on non-V5 platforms downstream

The MQSeries Application Programming Ref. (e.g. At http://www.software.ibm.com/ts/mqseries/library/manuals/) has the detail about sending to distribution lists in V5 and the mechanics of getting messages.
Response

Thanks *very* much for your informative answer. In addition to coming up with a workable scenario, you helped clarify in my mind some of the ways MQ works. A couple of questions, however:

Are distribution lists the same as name lists? If so, it looks (in my copy of the pubs) like this feature is only supported on MVS/ESA, not on all platforms. I need this solution to be platform independent, if at all possible. Does this change your scenario?

The scheme I had come up with was similar to yours, except I had the observers manage the distribution lists. I like yours much better, if I can use distribution lists on any platform.

Answer

Distribution lists and namelists are different things. Try looking at the current version of the manuals, available from


For example, distribution lists are supported on AIX and NT but not MVS.

Again, there are several possible approaches. One that would enable you to run your observer application on MVS but still use the MQSeries V5 distribution facility is to write the observer as an MQSeries Client application using the MQSeries Client for Java -- the client application can then connect to a version 5 queue manager on AIX/NT/etc. somewhere in the network. In fact, hundreds of observer apps running on different platforms could talk to a single queue manager located centrally. The pool queues that the pool applications read from could be resident on this same queue manager or downstream on different systems.

By using Java, you also get total platform independence; your app would run unmodified on Unix, Windows, OS/390 and other systems; as an applet, it would run in Netscape or other browsers on pretty much any platform. The only limitation here is that the MQSeries Client for Java needs TCP/IP to communicate with the queue manager.

Note: The MQSeries Client for Java, is available today as SupportPac MA83.

Note: There is a native (non-Java) MQSeries Client for MVS/ESA, but it doesn't support Version 5 functionality.

When using the MQ client interface, your application can run on one platform but use the services of a queue manager on another platform; note that for the client app to run, there must be an available network connection -- TCP/IP in this case -- between the machine where the client runs and where the MQSeries queue manager sits.

If this all sounds too complicated, you could forgo the MQSeries built-in distribution facility and just build a regular app on MVS that uses the local queue manager and distributes multiple copies of each message 'by hand' -- it's just a bit more work at the application level.

Can different platform processes send messages to the same Q?
**Question**

Is it possible to have two different platform processes send messages to the same queue, say one OS2 process one MVS process. The process that issues MQGET is on OS2 machine. For the MVS process, we set the the format as FMT_STRING prior sending the message. ON our OS2 receiving message side, we set MQGMO to MQ_CONVERT. what would happen, if the message is from the OS2 sending message process? If we don't specify FMT_STRING for the OS2 process when sending message, is this going to work?

**Answer**

There is no problem sending messages from different platforms to the same queue. There are no queue or message descriptor attributes which could restrict messages to coming from only one platform. Indeed, in a production environment it is probable that messages from many different platforms will be on the same queue.

You are correct in saying that if you specify the format as MQFMT_STRING on the MQPUT on MVS, and specifying MQGMO_CONVERT on the MQGET on OS/2, data conversion will occur (providing the other conditions for data conversion are met). The default format is MQFMT_NONE, and if this is detected on the MQGET call, then data conversion does not occur. This may not be a problem for the messages you have put on OS/2, provided the CodedCharSetId and Encoding fields on the OS/2 queue manager you put the message on are the same as those on the OS/2 queue manager you get the message from (the Encoding is not going to be different, but the CodedCharSetId may be). However, if you are going to send messages between OS/2 queue managers with different CodedCharSetIds, you should make sure the format doesn't default to MQFMT_NONE.

Chapter 11 of the MQSeries APG (-06 version) discusses data conversion in detail.

**Channel initiator for OS2 and AIX**

**Question**

Is the following correct? "I only need to use the runmqch command if I want to trigger channels"?

**Answer**

The Channel initiator(s) perform three tasks:

1. Starting the appropriate Channel when a trigger message is received on the initiation queue.
2. Retrying Channels whenever a Channel terminates abnormally subject, of course, to the Channel definition retry values.
3. Providing a base process for initiating Channels running as threads. Obviously a Channel running as a thread must have a process to be a thread of. Rather than a separate process the current products run the Channels as threads of the initiator.

So if you want any of these features, automatic starting of channels, automatic retrying of Channels or outbound Channels running as threads then you must have one or more Channel initiators running.
**CHL types for Switched Lines**

**Question**

I need to establish MQ connections between OS/2 PCs and MVS. The PC users will always decide when they dial to MVS, to optimize the communication costs.

So the users:

- dial
- start a CHL from PC to MVS

Then a CHL from MVS to PC should be started automatically, (or by the PC users).

What type of Channels do you recommend for this configuration?

**Answer**

This looks like an classic case of either requester/server or requester/sender depending on whether you want security or not i.e. On the MVS machine control who you send messages to. If you want security choose requester/sender.

**COMMANDS from an Application Program**

**Question**

We are in the process of developing a variety of clean-up/ housekeeping routines in some of our MQ application programs. There is a need to minimize the message traffic to the Command Input Queue but have found that I need to put each command as a separate message.

On looking at the relevant documentation I believe that I cannot string multiple commands into a single message OR use something like - DELETE QL(RODGER.TEST.*) and have the mqm delete all queues with the HLQs of 'RODGER.TEST'.

Is there something I might have missed or is there a command delimiter that can be used within a message to the Command Processor?

**Answer**

No you have not missed anything. You can only issue one command per message put to the SYSTEM.COMMAND.INPUT queue (See MQM MVS/ESA System Management Guide SC33-0806-00 - Building a message that includes MQM commands - page 155). There is NO command delimiter hence you cannot concatenate more than one command into a single message.

You can not issue generic DELETE requests. To avoid accidental deletes, you have to specify explicitly the name of the queue to be deleted. Hence, you have to issue a DELETE command for each queue to be deleted.
Syncpoint and Commit

Question

If a series of MQPUTs with syncpoint are made and then an MQPUT without syncpoint is made, does this cause the first series to be committed, does it change the whole session to nosyncpoint or is there another result?

Answer

There's another result! The first set will remain within syncpoint i.e. they will not be available for MQGET until MQCMIT. The message MQPUT without syncpoint will be available immediately for MQGET.

Cross Platform communications - creating and sending messages

Question

Once we have all the Qmgrs, queues, channels, etc defined on two machines(os/2 and AIX) how can we send a test message from one to the other without writing a pgm? How can we see this msg on the other?

Answer

Try AMQSPUT and AMQSGET programs. They are in mqm\tools\c\samples\bin.

Dead Letter Queue

Question

Can a Dead Letter Queue be defined as a Trigger Queue? I see nothing in the Command Reference that precludes this, but...

Answer

I assume that the question is if you can set triggering on on a Dead letter queue and the answer is yes you can. So you will have a trigger message sent to the Trigger queue when a message is put on the Dead letter queue.
Dead Letter Queue

Question

What are the consequences of having or not having a dead letter queue?

Is it better to not have a dead letter queue because of following reasons:

- When message can not be delivered, the message will remain in the transmission queue and channel is stopped
- If a dead letter queue is implemented, a handler need to be developed. The handler must sort out for what application the message is for and need to take proper actions. This is hard work. Without the implementation of a DLQ there is no need for such a handler.

Therefore, some questions on using MQSeries without a DLQ.

1. Is it wise to work without a DLQ
2. Is it possible to work without DLQ on all platforms (OS/2, MVS, DEC VAX and TANDEM)
3. Recommendations.

Answer

One reason for having a DLQ is to minimize re-transmissions of undelivered messages. For example, when putting messages to a remote queue, if the receiving MCA cannot put the messages to the target queue (because the queue is full say, or temporarily inhibited for puts (both of which can be regarded as transient situations), it will put them to the local DLQ. Therefore, avoiding the need for re-transmission by the sender, at some later point in time (if the DLQ is full or not defined, then in this case, the messages would be retained on the original transmission queue). Though, as you correctly point out, a DLQ monitor would now be required to process these DLQd messages.

Lets assume I only have one sender channel defined between two Qmgers, so that only one transmission queue is being served. Now lets assume that I have 3 applications putting messages to 3 different remote queues (each naming a different target queue on the remote queue manager but each naming the same transmission queue). If the receiving MCA finds that one of the target queues is full, then allowing the channel to stop (by not defining a DLQ) would cause a build of messages on the transmission queue. If a DLQ is defined however, the channel can continue to transmit messages.

MQSeries for MVS/ESA recommends that you name a DLQ though this is not mandatory.

Many customers do run without a DLQ, for the reasons you outlined. They seem to find it acceptable. It would be inadvisable if you expected to have undeliverable messages as part of normal running - for example if you had synchronous replies flowing back to a temporary dynamic queue which you deleted after a timeout. In that case late replies returning after the queue had been deleted would cause the channel to end. I do not believe any of the MQ implementations requires you to have a DLQ. In practice most customers with a DLQ never see anything in it in production, once they have sorted routing problems early on in test.

Again, remember that a channel will be stopped WHENEVER a message cannot be delivered by a receiving MCA (either no dead letter queue or the dead letter queue is not available). You will have no indication on WHAT message caused the interrupt and why.

The importance of a dead letter queue is much smaller in a local MQ environment with no DQM because name resolution is synchronous and the sender application will be informed directly. This is no more possible in DQM where the sender is already disconnected after putting the message successfully into a transmission queue.
I agree that the FULL handling of dead letters at EVERY node can become quite complicated. So I would suggest that dead letters that cannot be resubmitted locally will be sent to a central server via a safe stable channel for trouble shooting and logging/controlling.

**Default persistence when using ReplyToQ**

**Question**

Assume I have an incoming message with mqmd.ReplyToQ "MY_Q" and mqmd.RepyToQmgr "MY_Qmgr", and my server appl MQOPEN's with mqod.ObjectName "MY_Q" and mqod.ObjectQmgrName "MY_Qmgr". Also that I have a transmission queue "MY_Qmgr", with DEFPSIST(YES), and that I do MQPUT with mqmd.Persistence MQPER_PERSISTENCE_AS_Q_DEF.

In this case, is it the DEFPSIST of the transmission queue which determines the persistence of the message, or something else?

**Answer**

Your server application will be opening a fully qualified remote queue (ie: there is no physical definition for the remote reply to queue on the server), so the only object definition available is the transmission queue and, on MVS at least, we would indeed use the default persistence of the transmission queue when putting the reply message.

**Default transmission Queue**

**Question**

I understand the concept of a default transmission queue, but does anyone know if there is a way of defining a default QMANAGER to route to if the QMANAGER that an item is being put to is not known to the local queue manager, i.e.

```
Queue Manager SYSTEMA ------- -> Queue Manager SYSTEMB
```

Application on SYSTEMA
PUTs an item to SYSTEMC.QUEUE
where SYSTEMC is an MQM connected to SYSTEMB

As far as I understand, an MQM system has to have a definition for a QUEUE MANAGER to be able to place it on to a transmission queue (be that transmission queue default or explicit).

Am I wrong in this assumption, and can I setup a definition that will allow me to PUT items to queues that one system does not know about to the transmission queue of another system to see if that system can resolve it. If not then is always an overhead of adding a definition to an MQ manager if more and more downstream queue managers are added.
**Answer**

If an application on SYSTEMA puts a message to SYSTEMC.QUEUE where SYSTEMC is an unknown XmitQName, then the message will be placed onto the default XmitQ if it exists. The default Queue Manager to route to is therefore the QM on the other end of the channel which services the default XmitQ

**Design question**

**Question**

Scenario:

My customer has a program (PrgA) which puts a message on a queue (ReqA) and receive its answer from the message queue (AnsA). This looks very simple if only one user uses PrgA at a time. But what happens if several user use PrgA? Since they all use the same queues ReqA and AnsA, how could the PrgA distinguish, which message belongs to its user?

- Should every user have its own queues for each application? How to define these queues (local or modal) and how to handle them (since there might be hundreds of them)?
- Should it be in the PrgA logic to distinguish between the users? But how to design the PrgA? Looping in the reply queue (with hundreds of messages) for one particular answer for a user does seem to be efficient, or?
- What basic idea of MQM did I misunderstand or did I miss?
- Do you have any design tips for the PrgA o the queues?

**Answer#1**

This need is solved by using MSGID and CORRELID.

If the MQGET program puts the reply with the msgid and correlid from the first message, then PrgA can get the reply using the same msgid/correlid.

It's like a keyed read - you can go straight to the messages that are for you - if the answer has arrived yet.

You can request that MQ create a system determined, unique msgid, by specifying MQMI_NONE as the MSG_DESC.MSGID, and your application creates the correlid - application name, for instance - in MSG_DESC.Corrrelid. (Where MSG_DESC is a MQMD). A correlid may not be needed.

The same msgid can be specified in subsequent MQPUTs, if it is copied from the MQMD after the first MQPUT has completed.

**Answer#2**

There are two usual ways to handle this:

1. All instances of PrgA share a common reply-to queue.

   In this case, each instance of PrgA remembers the message id of the request it placed on ReqA (this is returned by MQPUT). It then uses MQGET on queue AnsA with a correlation id equal to this remembered message id.
This MQGET will only complete when an answer message with this correlation id arrives in queue AnsA - answers for other instances will be ignored. This assumes that the application serving ReqA behaves "normally" and copies the message id from an incoming request to the correlation id of an outgoing reply.

2. Each instance of PrgA has its own unique reply-to queue.

In this case, each instance of PrgA opens a model queue for its reply-to queue (AnsA). This causes a dynamic local queue to be created with a unique (system-generated) name. There is no need to remember the message id, as the only reply messages arriving on the dynamic queue will be those for this specific instance of PrgA. Assuming that the messages used are non-persistent (i.e. do not need to be recovered if the queue manager is restarted), the model queue can specify that the dynamic queue should be temporary. This has the advantage that the dynamic queue will be deleted when the application ends. Again, this assumes that the server program behaves "normally" in that it gets the name of the reply-to queue from the incoming request message (rather than having it hard-coded within the application).

---

**Exact size of the message header**

**Question**

What is the exact size of the message header in V2 products?

**Answer**

There are two headers you need to consider for resource planning:

1. The message descriptor (MQMD) which is stored with every message.
   - The MQMD is 324 bytes long.
   - This is the number you need when planning DISK capacity for APPLICATION queues.

2. The transmission queue (MQXQH), which includes the MQMD. It is stored with messages destined for remote queues and travels across the network with the message.
   - The MQXQH is 428 bytes long (104 + 324 for the MQMD).
   - This is the number to use when planning DISK capacity for TRANSMISSION queues and for NETWORK performance/capacity.

The headers are documented in detail in the MQSeries Technical Reference (SC33-0850) and Application Programming Reference (SC33-1673) manuals.

If you are concerned about the network overhead on short messages, or if you have slow lines, you may want to consider compressing the messages while they are on the network. Potential solutions include:

1. Compression options built into some network types.
2. SupportPac MO02: download from the MQSeries web-site.

There is one other number that may have prompted the original question, namely the constant MQ_MSG_HEADER_LENGTH documented under the description of the "BufferLength" parameter of the MQPUT call.

You may be aghast to see that MQ_MSG_HEADER_LENGTH has the value 4000, but rest assured -- the MQ headers are nowhere near as big as this. The reason for this constant being so big is to ensure that applications
which put messages as large as possible continue to work in the future when the MQ headers grow a bit. The application is recommended to put messages no larger than the queue's "MaxMsgLength" attribute MINUS the value of MQ_MSG_HEADER_LENGTH. This results in some spare space into which the MQ headers could expand in the future, if needed.

**EXPIRY**

**Question**

Which are the conditions under which msgs put on a queue with an expiry time already reached, are deleted from a queue? On a MQM/ESA 1.1.4 I'm not able to eliminate these messages, or better to reset the DEPTH COUNTER.

**Answer**

Expired messages are never "deleted" by the Qmgr until some program (either application or MCA) attempts to get the message via the MQGET call. Only then will the message be physically deleted from the queue and any report messages generated. On certain platforms, the MQGET has to be a destructive operation, not a browse. In any case, expired messages are never given to an application by the Qmgr.

See the MQSeries Application programmers Reference (expiry field of the MQMD section).

**Full Xmit queue.**

**Question**

If a transmit queue becomes full for some reason (probably channel that serves it is down), then a application that is trying to issue a PUT to a local definition of a remote queue will get a reason code back stating that a queue is full. Which queue? I think - transmission queue. Is this correct?

**Answer**

Yes, only local queues hold messages. Since an open of a local definition of the remote queue resolves down to the transmission queue, if the transmission queue is full an application attempting to put to the local remote queue will receive MQRC_Q_FULL.

**Message channels and multiple connections**

**Question**

I have the following configuration using MQ SDR/RCVR channels.
MVS Qmgr <--------> AIX Qmgr

I need to add additional AIX workstations that can send/receive messages from the MVS Qmgr.

Can I do this with additional SDR/RCVR channels or do I need a client/server channel setup?

Does a channel initiator only manage 1 set of sdr/rcvr channels?

Is it possible to have a 2nd CHINIT on MVS to the same Qmgr and define SDR/RCVR channels that it manages?

I read through some of the Dist. Queueing Guide and it seems to me you can only connect 2 Qmgrs unless you have a client/server setup.

**Answer**

You can define as many channels between your local Queue Manager and other Queue Managers as you wish (within system limits). Likewise, a single Channel initiator can handle as many Channels as you like. On MVS you can have only one initiator, on other platforms you can have many of them although it is very unusual to require more than one.

Each Channel pair can be defined as whatever types make sense for that particular connection. The most commonly used types are SDR/RCVR. Therefore on you MVS machine you can define :-

- A transmission queue to each of your AIX machines (or any other machine)
- A sender channel to each AIX machine
- A receiver channel from each AIX machine

The channel names must obviously be unique.

**Message Prioritization**

**Question**

Let us say we have three messages on a queue with priorities of 2, 3, and 7.

```
237 Queue
```

We then do an MQPUT on a message with a priority of 8. My curiosity is the mechanisms of the MQPUT on a message with this high priority.

What does the queue look like after the MQPUT? What does the queue look like before the MQGET?

Does the queue look like the following after the MQPUT:

```
2378 Queue
```

or is it:

```
2378 Queue
```
where a search is done on the MQGET?

Answer

Messages are inserted into the queue by priority. In your example, the queue initially had messages with priorities 2, 3, and 7. Those messages would be on the queue with priority 7 at the head (first to MQGET) and priority 2 at the tail (last to MQGET).

A new message of priority 8 would be inserted at the head of the queue, in front of the priority 7 message.

The queue manager maintains pointers to the first and last message (or page of messages) of each priority level for each queue, making it possible to locate the insertion point without searching through messages.

MQGET traverses the queue from head (front) to tail (back) looking for the first "eligible" message.

Eligibility is mostly based on locking (is the MQPUT that inserted the message committed) and sometimes by MSGID/CORRELID values.

MQGMO_BROWSE_NEXT sometimes does not process a message

Question

AIX MQ, I have a persistent queue to which many processes put requests, only one process gets those messages sequentially with MQGMO_BROWSE_NEXT option. I noticed that some messages left in the queue, never processed. With MQGMO_BROWSE_FIRST option, all messages are processed no stalled messages. It seems that I should always use MQGMO_BROWSE_FIRST option.

My question is when should I use MQGMO_BROWSE_NEXT option instead of MQGMO_BROWSE_FIRST option?

Answer

When an application is browsing a queue, with (say) the browse cursor somewhere in the middle of the queue, there are two circumstances when new messages can arrive PRECEDING the browse cursor (i.e. they appear on the queue somewhere between the start of the queue and the browse cursor):

1. The queue is a PRIORITY queue and a message arrives with a priority greater than that of the message the browse cursor is on.

2. There was an uncommitted unit of work which caused a message to be placed at the end of the queue, but other messages arrived and were committed before that unit of work. The message in that unit of work does not
become visible until the unit of work is committed. Consequently, if the browse cursor has already passed over it by the time the unit of work is committed, the browse cursor will not see it on that scan of the queue (using MQGMO_BROWSE_NEXT). This happens for FIFO queues, as well as for PRIORITY queues.

Case (1) above can be avoided simply by defining the queue to be a FIFO queue. But case (2) cannot be avoided if messages are placed on the queue within units of work.

The solution lies with the application design. When the browse cursor reaches the end of the queue, the application should reset the cursor to the start of the queue (by using MQGMO_BROWSE_FIRST) in order to verify that no new messages have arrived. This can be awkward if the application did not process and remove all of the messages it previously browsed, since now it will scan them again. If the application REALLY wants not to process some messages on the queue, it would have to keep a record of the ones it had already scanned, so that when doing a rescan it could tell whether any new messages had arrived; if none had arrived the application should (presumably) terminate rather than scanning the queue for yet a third time!

If the application ALWAYS processes the message it has just browsed, then it must always be processing the first message on the queue, and so in that case it can always use MQGMO_BROWSE_FIRST instead of MQGMO_BROWSE_NEXT.

MQI channels vs. Message channels

Question

Please distinguish between the two and when each is used. Which one is for remote communications? Any info on the underlying protocol i.e. connection usage, protocol handshaking, etc.?

Answer

Distributed Queuing Guide: 1.1.8 What is a channel?

A channel is a logical communication link. There are two different categories of channels in MQSeries, (with different channel types within these categories):

**Message channel** This connects two queue managers via message channel agents (MCAs), and is unidirectional. Its purpose is to transfer messages from one queue manager to another. A channel definition exists at the sending end of the link and at the receiving end.

**MQI channel** This connects an MQI client to a queue manager on a server machine. It is for the transfer of MQI calls and responses only and it is bi-directional. A channel definition exists for each end of the link and there are different ways of creating and using the channel definitions (see MQSeries Clients book for more information).

Channel definitions, of both categories described above, must include a channel type as well as a channel name. You can choose to use different channel types according to the application you are designing, but the same channel name must be used at both ends of each combination.

MQI Client MQCONN returns code 2059
**Question**

I am trying to use OS/2 MQI Client with simple channel using

'SET MQSERVER=TARGET3.SVRCONN/TCP/TARGET3'. On the server I have

defined a SVRCONN type channel (name = TARGET3.SVRCONN) with TCP protocol. My client application gets code 2059 following MQCONN call to 'target3.queue.manager', which is the queue manager on TARGET3 machine. I TCP/IP ping TARGET3 successfully.

Am I doing something wrong? I have looked at the error and saw the following entry:

  AMQ9202: Remote host 'TARGET3 (9.11.1.111) not available, retry later.

  The return code from TCP/IP is 10061 (X'274D').....

Any suggestion would be appreciated,

**Answer**

Do you have a Listener or inetd listening on port 1414 on your server machine? The message is trying to tell you that the client tried to contact the server by no one answered.

---

**Not a Valid Trigger Message**

**Question**

We have set up triggering on the queue manager so that the arrival of a message in a particular queue will trigger a process. When the message arrives however, the MQM system displays a message "Not a valid Trigger Message" I understand they the MQmanager actually generates the Trigger Message. How can it not be valid?. There is no message number associated with this error.

**Answer**

It sounds as though you are putting the messages directly to the initq that the trigger monitor is watching rather than to the queue for which they have triggering switched on.

---

**Performance Tuning**
Question

We have the following configuration in a test environment:

MQClient (Windows 3.1) <=> MQServer (OS/2) <=> MQServer (MVS)

We installed all of the queue managers using the default values. After writing some applications, we would like to know what can be done to increase performance for our test users.

1. Are there any tools for performance analysis available for the different platforms? How can we figure out where most of the time is spend?

2. Where should we start or where are normally the bottle necks?

3. Which time delays for getting the requested answers are acceptable or "normal"? When does it make sense to analyze the performance?

Answer

The following hints can generally improve MQ performance:

1. Ensure MQ log files and queue files are on separate disks.

2. Use a high batch number on your channels. You will see a big improvement in changing this from 1 to 50 then a smaller improvement in changing it from 50 to 200.

3. Arrange that the application keeps the transmit queue supplied with messages. When the transmit queue depth falls to zero the messages are sent no matter what the batch number is. At this time you experience the end of batch processing.

4. If possible make the size of the message you are sending over a TCP/IP channel longer than 500 bytes long and shorter than 3596 bytes long. MQ adds a 500 byte header and TCP/IP is slower for messages shorter than 1000 bytes or longer than 4096 bytes. (see AIX performance and tuning guide SC23-2365-03 page 8-10).

Priority of event messages

Question

Can different priorities be set for specific MQ generated events?

Answer

No, you can not allocate specific priorities to selected event msgs destined for the same event queue. Event msgs take the default priority of the event queue. So, an event msg with reason code MQRC_Q_MGR_ACTIVE will be placed on the SYSTEM.ADMIN.QMGR.EVENT queue with the default priority of this queue. Clearly, this means that other event msgs placed on this queue will also be placed with the default priority.

However, you can set different default priorities for the three types of event queues.
Question

We have several applications which are PUTting messages to remote queues and we want some of them to be delivered first and some of them to wait if important ones are coming.

Does a message with priority 3 for example have to wait already in the transmission process (maybe in the XMIT queue) if there comes a message with priority 1?

So is using message priority a way to help his important messages be delivered more quickly over the network than less important ones. We know there is the possibility of having separate channels and XMIT queue but then he has a lot more definitions (he has about 150 servers).

Answer

Firstly, priority 9 is the highest message priority and priority 0 is the lowest message priority.

The order of delivery of messages is determined by the value of the local queue attribute MSGDLVSQ. As transmission queues are local queues, you can set this attribute on your transmission queue. If you set this value to FIFO, then messages will be delivered (in response to MQGET calls from the queue) in the order in which they arrive on the queue (ie: the priority of the message will effectively be ignored). However, if you set this value to PRIORITY, then messages will be delivered (again, in response to MQGET calls from the queue) in PRIORITY order in fifo (ie: priority 9 messages will be delivered first in put time order (oldest first), followed by priority 8 messages, then priority 7, etc. Etc.).

So, if you have priority 2 messages on your transmission queue and some priority 8 messages arrive on the transmission queue, then provided MSGDLVSQ is set to PRIORITY, the sender MCA will retrieve, and hence send, the priority 8 messages before the priority 2 messages.

PTF Download

Question

Where can I go to download a PTF or CSD?

Answer

PTFs and CSDs can be downloaded from:


Queue Manager Hierarchies for enterprise networks
Question

We are currently designing an enterprise wide client/server system based on MQM (host and workstations). We have many server systems with a separate queue manager. We want now that each server can communicate with each server in the network. We want to avoid MANY-TO-MANY queue manager connections, so we are thinking about a hierarchical organization of the queue managers to keep administration simple (like TCPIP name servers). Could somebody give us a direction, how MQM supports routing through large networks? Is there a general approach to this problem?

Answer

The simplest way to set up a hierarchical network is to make use of the default transmission queue. If a queue manager does not have a definition of how to get to a particular queue manager it will put the message on the default transmission queue. If you make the default transmission queue point to the queue manager that is upwards in the hierarchy then messages will naturally percolate upwards to the top which must have the routing information.

```
    A
   / \
  B   C
 / \ / \
D   E   F   G
```

The default route for each qmgr points to the qmgr above it in the hierarchy. eg D passes unknown queue manager messages to B. B passes them to A.

Each qmgr must know about the qmgrs below it in the hierarchy. E.g. B knows how to get to D and E. A must know about all the queue managers. B does not need to know about C, F or G.

A hierarchical network is the simplest to define and reduces the number of queue manager alias definitions needed to define routes. Unfortunately it does not reduce the remote queue definitions that are needed to discover which queue manager owns a particular queue.

You might like to think about how you name queue managers. It might make sense to start putting some structure in the names which might allow simpler routing definitions. Eg D might be called B.D

Queue priority

Question

Assume there are multiple queues and messages are put into queues while network link is off. When the network comes back, MQ will try to deliver these messages in queues. The question is that which queue will be first served? What is the rule? Can a user specify which queue goes first?
Answer

The transmission queue is a local queue. Associated with a local queue is a MSGDLVSQ attribute which can be set to FIFO or PRIORITY. If you set it to PRIORITY, then this will ensure that your higher priority messages are delivered first. So, if you get the situation where a queue fills up, say with 2 priority 9 messages and 1000 priority 3 messages, while your channel is down, then the 2 priority 9 messages will be delivered ahead of the 1000 priority 3 messages.

If you have multiple transmission queues, as each transmission queue is served by a different channel (which has it open for input exclusive), when you start the channels, the messages will start to flow together. I am afraid that the concept of allocating priorities to channels, so as to control the speed at which channels retrieve messages from transmission queues, does NOT exist.

Remote Q and the DeadLetter Q

Question

We were just told we would be using MQ to communicate between AIX and MVS/IMS. Been reading up on this stuff and it looks to be not so hard, but I am a bit confused about the Deadletter Q.

Let's say we are sending messages from SystemA to SystemB - all datagram types - no request/response needed, but do need guaranteed delivery and guarantee that messages arrive in the same sequence they are sent.

What happens if the link goes down? Are the messages now saved in local representation of the remote Queue? Will they only go into the Deadletter Queue if this local Q fills up (or some other really bad error). Or do they immediately go into the Deadletter Q because they could not be delivered to the remote Queue.

Now when the link comes up again, the Queue Managers start exchanging data again, but the ones in the DeadLetter lay around until I process them - thus messages arrive at remote system out of order.

Answer

If the link goes down the messages do not go to the dead letter queue. The messages you send to a remote system are always placed on a local queue (transmission queue) ready to be sent to the remote system by a channel servicing that transmission queue. The fact that the communication link is not available or that communication is lost does not matter. The messages remain on the transmission queue until a connection can be made to the remote system. So, the messages will be delivered in the correct order.

Message only go to the dead letter queue as a last resort - if, for example, it is discovered that the destination queue does not exist. The dead letter queue is for messages which could not possibly be delivered.

Rules of thumb
Question

I am a new to MQSeries and I have some general questions and I am looking for some rules of thumb. In particular, I am looking for some conventions or standards on:

1. number and names for queue managers
2. naming conventions for other queue objects
3. standard configurations for communications between AIX and MVS/CICS

Answer

We have an internal working group which has been looking at this area. Here is a summary of some of our ideas.

1. QUEUE MANAGER NAMES:

   This can be influenced by platform. If you have an MVS (CICS) Qmgr as one of the members of your network of Qmgr's, then that one has to have a four character Queue manager name. This is very limiting. If you limit yourself to one QMGr on one MVS system, then you could use an MVS entity such as the SMF id as your unique four character Qmgr name. Or you could say the first 3 relate to the identity of the MVS system, and use the fourth to allow 2 or more. A good rule of thumb is to make all your Qmgr names unique, whatever naming convention you adopt. Then in the future if you need to setup channels between 2 Queue Managers you thought would always be independent, then you won't get caught out!

   Also, if you are using a platform that allows longer Queue Manager Names don't use all available characters just because they are there!

2. NUMBER OF QUEUE MANAGERS:

   You can almost have as many as you want. In practice, you should plan on a modest number. Once you have more than one on a machine, then whatever the platform is, you need to consider which one is the DEFAULT Qmgr. You could easily find a pitfall of work going to a (wrong) default Queue Manager, if you don't think about this.

3. CHANNEL NAMES.

   You could do a lot worse than:
   
   CH.<Qmgr-A>.TO.<Qmgr-B>

   Remember that the SDR at one end has to have the same name as its partner RCVR at the other end. I guess SVR/RQSTR pairs would have to match too, but I haven't done those.

4. QUEUE NAMES.

   You could have these prefixed by project name,
Question

I have defined a remote queue and some channels using TCP/IP. From what I can tell, you must execute runmqchldata actually sent to the other system. And it seems like runmqchl executes a program out of inetd on the remote system and passes data via sockets. A couple of questions:

1. runmqchl eventually times out and goes away. I assume this is as designed, but I am curious how this should be configured or use if people are sending data to the remote queue all day long. Do I need to write a shell script to restart runmqchl if it times out?

2. Is there a way to automatically execute runmqchl only when there is something to send?

Answer

The channel process will terminate after the disconnect interval, defined by the DISCINT parameter on the channel definition, expires. You can set this value to 0 so that the channel never disconnects but it is better to do what you want to do and only run the channel when there are messages to send. This is possible using the channel initiator program. To make this work you need to define the transmit queue so that it is triggered.

In detail, what you need to do is as follows.

1. Define a local queue to be used as an initiation queue. (This queue must be defined NOTRIGGER) or use the default SYSTEM.CHANNEL.INITQ

2. Define a process with your channel name in the USERDATA field.

3. On the transmission queue for the channel you must define the following
   
   TRIGGER
   TRIGTYPE(FIRST)
   TRIGDPTH(1.
   PROCESS(as defined in step 2)
   INITQ(as defined in step 1.

4. When you start the queue manager you must also start a channel initiator using
   
   runmqchi -q initq -m Qmgrname

Then the channel will be started automatically whenever there are messages on the transmit queue.

Incidentally, you also need to be running a channel initiator if you want channels to try to reconnect after a failure.

Security Exit

Question

In the Distributed Queue Management Guide under Security exit, it says that ‘this end may initiate a security message exchange with the remote end by providing a message to be delivered to the security exit at the remote end...’. Can some one explain how this is done. We are thinking of using this function to exchange some keys with the remote end for security implementation.
ON MQM for OS/2 and AIX, there is a field called security exit user data under channel definition. Is this the field used for the above purpose for passing data to the remote end? But I don't seem to find the same field under MQM MVS/ESA?

**Answer**

The security exchange data is passed to the remote end using the AgentBuffer or, if you choose, the ExitBufferAddr parameter described in the MQCHANNELEXIT call definition (page 211 of the DQM Guide, SC33-1139-02). The SecurityUserData parameter of the Channel Definition is used for passing data to your security exit program. There is nothing to stop this being your intended security message (if your exit program copies it to the AgentBuffer). However, it would be highly ill-advisable because any user who can display the channel definition could then see your security message. Instead, your security exchange message should be defined within your security exit program only, and then use system security (eg. RACF) to stop unauthorized users getting access to the program.

In answer to your final question, the exit UserData fields are not available on MQSeries for MVS/ESA if you are using DQM via CICS. However, they are available in the recently announced MQSeries for MVS/ESA V1.1.3 product if you are doing DQM via TCP/IP or APPC.

The following provides some greater detail on how you could code your security exit program(s).

Usually, you would have a security exit program defined at both the sender and receiver ends of the channel. These programs would reach agreement that the partner end of the link is genuine or else they would close the channel. Figure 110 on page 202 of the DQM Guide (titled 'Receiver-initiated exchange with agreement') outlines a typical flow. Here, both the receiver and sender exits are initially invoked with ExitReason MQXR-INIT in the MQCXP. Your exit program would test for this, and respond by setting the ExitResponse field of the MQCXP structure to MQXCC-OK. Next, the receiver channel is always given the initial option of exchanging a security message when it is invoked with ExitReason MQXR-INIT-SEC. It may decline to do so by setting ExitResponse to MQXCC-OK (in which case the sender security exit will then be invoked with ExitReason MQXR-INIT-SEC), or it may accept by setting the ExitResponse to MQXCC-SEND-SEC-MSG or MQXCC-SEND-AND-REQUEST-SEC-MSG. In this case, it would pass the security message it wishes to send in the AgentBuffer and set the DataLength field to the length of the security message.

The receiver MCA would then send the security message to the sender MCA. The sender MCA would then invoke the sender channel's security exit with ExitReason MQXR-SEC-MSG and pass the security message to the exit in AgentBuffer. The length of the security message would be in DataLength. The sender channel's security exit should check the content and length of the security message is has received are as expected. If they aren't, the exit program would presumably request closure of the channel by setting its ExitResponse to MQXCC-SUPPRESS-FUNCTION (I say 'presumably' because what the program does is entirely up to it - the MCA won't close the channel unless the exit tells it to).

However if the sender channel security exit is happy with the security message it received, it can respond in two ways:

1. Set ExitResponse to MQXCC-SEND-SEC-MSG or MQXCC-SEND-AND-REQUEST-SEC-MSG, send a security message of its own via AgentBuffer or ExitBufferAddr, and set DataLength accordingly. This would then cause the receiver channel security exit to be re-invoked with ExitReason MQXR-SEC-MSG. The receiver channel security exit could then process this security message, and respond as it sees fit.

2. Set ExitResponse to MQXCC-OK. This indicates the security exit is happy, but that it has no security message to send back to the receiver channel security exit. This will cause the receiver channel security exit program to be re-invoked with ExitReason MQXR-SEC-MSG but with a DataLength of zero (this is done because unless one of the security exits requests closure of the channel, the security exchange must complete at the side which initiated it; thus if a security exit is invoked with MQXR_INIT-SEC and it does initiate an exchange, it will
always be next re-invoked with MQXR-SEC-MSG). When the receiver security exit receives the zero length
security message, it may decide this is unacceptable and close the channel (for example, because it had initially
set MQXR-SEND-AND-REQUEST-SEC-MSG), or it may be OK. In this latter case, the receiver channel
security exit program would set its ExitResponse to MQXCC-OK. At this point the security exchange is
complete.

Once the security exchange is complete, any user messages can begin to be sent down the channel (assuming you
have no other exit programs to invoke).

As you will have seen, most of the flow is determined by how the MCA sets the ExitReason field of the MQCXP,
and how your exit program responds via the ExitResponse and ExitResponse2 fields of the CXP. The contents and
length of the security messages are passed via the AgentBuffer/ExitBufferAddr and DataLength fields described by
MQCHANNELEXIT respectively.

Self Study Guide

Question

Is there a MQSeries Self Study Guide or something which is similar to that besides the Reference manual?

Answer

MQSeries CBT

This 12 hour self study computer-based training course describes the MQSeries* of products, their functions, and
use. The course is designed both as an introductory course in its own right, and as a prerequisite course to other
more technically oriented courses. Customers and IBM* technical personnel who want to find out more about
MQSeries and its queue managers should benefit from this course.

For additional information, you can access a one page flyer by calling 1-800-IBM-4FAX (415-855-4329) and
requesting document number 7587.

COURSE DETAILS for MQ900:

After completing this course, you should be able to:

- Define and describe the various types of program-to-program communication
- Differentiate messaging and queueing from other methods
- Define and describe the structure and function of a messaging and queuing system as implemented by the
  MQSeries products
- Describe the fundamentals of programming the MQI
- Describe the basic Message Queue Manager (MQM) administration and operation
- Describe Queue Manager interoperation
- Identify typical MQ* applications

TOPICS INCLUDE:
- Messaging and queueing (MQ) architecture and operation
- Programming the Messaging and Queueing Interface (MQI)
- MQ applications

PREREQUISITES:

Prerequisites for this course include an understanding of basic computer concepts, knowledge of or experience in production computer environments, and knowledge of at least one of the following operating systems: MVS*, VSE, OS/2*, OS/400*, AIX* (or other Unix** variant), VAX/VMS, Stratus, or Tandem.

MACHINE REQUIREMENTS:
- 486DX at 66MHz or higher
- 16MB RAM, or more
- Microsoft**-compatible mouse
- 2X CD-ROM, or faster
- Sound Blaster-compatible sound card
- VGA 256-color graphics capability
- Microsoft Windows** 3.1, Windows 95, or Win-OS/2

COURSE MATERIALS:

Each program package contains:
- CD-ROM with program code
- Installation and deinstallation instructions
- IBM International Program License Agreement
- Proof of Entitlement

TO ORDER:

To purchase a copy of the CBT, call IBM Education and Training in your country and request MQ90 MQSeries Technical Introduction CBT

     Australia  1-800-801-088
     Austria43-0660-8814
     Denmark45-4523-3108
     Germany49-1805-5090 (01805-5090 when calling in Germany)
     Finland358-9-459-4444
     France 33-02-38-55-77-77
     Italy  39-167-017001
     Japan  81-44-201-5182
     Netherlands31-020-513-3940
     Norway 47-669-998-299
     South Africa  27-011-302-6619
     Spain  34-901-100-400
     Sweden 46-8-793-4408
     Switzerland41-01-643-69-69
     UK 44-0345-581329
     US and Canada  1-800-IBM-TEACH

If your country is not listed, call IBM and ask to be connected with the Education and Training department. When ordering MQ900, specify Feature Code #9001
Sender/Receiver Question

Question

A simple question:

We have a number of places where we have channels between Qmgrs - and up to now have set up SENDER/RECEIVER pairs.

When would you setup SERVER/REQUESTOR instead? Does this imply a Client Server relationship, rather than 'partners'?

Answer

Sender/Receiver is correct 95% of the time. Use Server/Requester when you need to start/stop the channel from the receiving end. For instance, in a dial-up connection you probably want to start the channels from the end that initiates the dial-up. Otherwise, stay with Sender/Receiver unless you have really strong reasons not to.

The length of Msg.

Question

We want to transport a file about 1M bytes via MQ. We can transfer it as one message or split it into several smaller messages. Which way is more efficient? What are the pros and cons of these two ways.

Answer

Pros for Transmitting large message

1. Will be faster in transmission
2. Far less overhead in Queue Manager and communications code
3. Less disk I/O
4. Less work for the programmer, needn't worry about reconstructing the original data in the correct order.

Cons for Transmitting large message

1. Ties up the channel for the whole duration of the message. Over a slow line this could mean no other messages get through for some time.
2. If there's a communications failure then the whole message must be retransmitted (Important if you have really noisy lines).
3. Uses more memory.

Pros for Lots of small messages


1. Allows Parallel processing If data can be processed sequentially, receiving process can start processing the data before it has all been sent.

2. Allows messages of a higher priority to have preference over message transfer.

**Cons for Lots of small messages**

1. More expensive in CPU and I/O

2. Possibility of some of the messages being re-routed.
Duplicate reason codes in MQ V5

**Question**

We are in the process of upgrading from MQSeries V2.2.1/AIX and MQSeries V2.0/NT to MQSeries V5, and have run into problems with duplicate reason codes. Specifically the MQRC_CONVERTED_MSG_TOO_BIG and MQRC_TRUNCATED reason codes.

In the version 2.x MQSeries products these were two separate reason codes (2120 and 2151 respectively). However in version 5.0 MQRC_TRUNCATED has been reclassified to 2120. The previous value (2151) is still available, but no longer in use.

Applications which check these values (in our case a case statement) will not recompile due to this conflict.

Why has MQRC_TRUNCATED been reclassified and will it be changed back or should it not be used?

**Answer**

This is the situation with regard to these two reason codes. In the 2nd edition of the "Application Programming Reference" (SC33-1673-01 December 1995, for the version-2 products):

- MQRC_TRUNCATED was described as being returned by the MQXCNVC call.
- MQRC_CONVERTED_MSG_TOO_BIG was described as being returned by the MQGET call.

Unfortunately, that is not what the implementation did -- the queue manager returned MQRC_CONVERTED_MSG_TOO_BIG in both cases. Regrettably this was not discovered until some months after the version-2 products were available.

The question then was, should we:

- Change the implementation to correspond to the book, or
- Change the book to correspond to the implementation?

The decision was taken to do the latter, and the version-5 documentation (SC33-1673-03 August 1997) now correctly describes the implementation. The reason code that should be used with both MQXCNVC and MQGET is MQRC_CONVERTED_MSG_TOO_BIG.

But what of MQRC_TRUNCATED? This is now an obsolete reason code that is not returned or used by any part of the queue manager. What should happen to it?

- We could have removed it from the header files, but that would have meant compilation failures for those applications that were referencing it.
- We could have left it in the header files with its original value. The disadvantage of that is that applications that used it would not be working correctly (they would be testing for the wrong reason code), and although they would recompile without error under version-5, they would STILL be wrong.
- Instead we chose to change the value of MQRC_TRUNCATED to be the same as MQRC_CONVERTED_MSG_TOO_BIG. This has the advantage that when applications that reference it are
recompiled, THEY WILL START WORKING CORRECTLY for the first time, whereas before they were not working correctly (because they were checking for the wrong reason code).

Unfortunately you have encountered a combination of circumstances where this change gives a compilation error:

- Your "switch" statement is testing explicitly for BOTH reason codes (which is perfectly valid, but somewhat unexpected).
- You are using a programming language that does not allow multiple "case" statements with the same value, namely C.

Of the three most-popular MQSeries programming languages (C, COBOL, and PL/I), only C does not permit duplicate "case" values; both the COBOL "evaluate" and the PL/I "select" statements allow duplicate values in their "when" clauses.

So in summary:

- On version-2 and version-5 queue managers, change all occurrences of MQRC_TRUNCATED to MQRC_CONVERTED_MSG_TOO_BIG.
- If duplicate "case" values result, delete one of the cases.
- On version-5 only, a recompile is sufficient if there are no duplicate cases (you don't need to modify the application source).

I hope that this clarifies the situation, and explains what changes you need to make to your application. We are sorry that this error occurred.

---

**Technique to keep channel up.**

**Question**

I experience frequent channel time outs. Is there a good technique to do the following:

1. trigger the channel for the very first message
2. for subsequent messages, trigger the channel IF (and only if) the the channel is not available (i.e. timed out)

and by the way: how can I set the timeout limit for the channel?

**Answer**

You have solved your own problem! Use triggering (supported on all level 2 platforms) on the transmission queue, with a trigger type of "first". This will generate a trigger message when a message first arrives on the transmission queue, but not for subsequent messages. Look at the Distributed Queuing Guide for details on triggering.

If you combine this with an appropriate channel disconnect interval (i.e. one that is slightly longer than the arrival time of messages on your transmission queue), then you will only use MQSeries and network resources when there are messages that require remote transport.
Response

I did just that. I set the transmission queue with a trigger that invokes a process whose user data field contains the name of the channel. I also ran the channel initiator when I started the queue manager. The first message seemed to wake up the channel alright. I then took down the channel (STOP CHANNEL) and tried to put a message to the remote queue again. This did not work because the system changed the trigger indicator to NOTRIGGER on the transmission queue and the message got stuck there. I am running MQS on AIX, and am not quite sure if it behaves the same way on MVS.

Answer

Don't issue the STOP CHANNEL. That, as you have observed, switches triggering off. Change the disconnect interval so that the channel stops of it's own volition.

The only way for a channel to end normally and become INACTIVE is if the disconnect interval on the sender channel expires. This will put the sender into INACTIVE state, from which it can be triggered again when new messages on the transmission queue. If you issue a STOP CHANNEL command or if the channel stops because of an error, the channel goes into STOPPED state, the xmitq is set to GET(DISABLED), triggering is turned off, and you have to issue the START CHANNEL command to restart the channel.

Chapter 4 of the Distributed Queueing Guide (-07 version) provides details on channel states.

Trigger Interval Value

Question

I am confused by Trigger Interval Parameter used by MQSeries (both ESA and NT environment). Is the value used at system level (queue manager) or is it attribute of a single queue?

Answer

TrigInt applies to the entire Qmgr, not a single queue. Basically, trigint applies to queues with a trigtype=first. Normally, the queue manager will only generate a single trigger message for this type of queue (ie, when the queue depth goes from zero to a non-zero value, when the initiation queue is first opened, or when the last application closes the queue and there are still messages on it.)

Trigint causes the queue manager to generate a trigger message when another message arrives on the application queue EVEN IF none of the above conditions apply. If Trigint seconds has passed since the last trigger message was generated for a specific queue, and another message arrives on the queue, the the Qmgr will generate another trigger message (even if trigtype=first).

Trigger monitor for MVS
**Question**

I have a Queue Manager 1.1.4, running on MVS, and I need to use triggering in a non-CICS and non-IMS environment. I downloaded the SupportPac MA12. The trigger monitor program runs, but the Queue Manager does not seem to put a trigger message on the defined initiation queue. I have set up the queue, process, and defined a initiation queue similar to a non-MVS platform MQSeries queue manager that is operational. The command server is running on MVS.

**Answer**

Double check all your definitions. Triggering on MVS is the same as on the distributed platforms. Remember that the trigger monitor must be running for the queue manager to create trigger messages. Are you sure that no messages are being created? Are you seeing the: 'Received Trigger Message; Generated JCL follows.' message on SYSOUT of CKTIBAT?

**Response**

Thanks for the help. I had set the queue to TRIGGER, but the trigger set to NO. I got the Queue Manager to trigger a message to an initiation queue called SYSTEM.INITIATION.QUEUE, and the trigger monitor picks up the trigger message.

**Trigger on last message**

**Question**

What's the best way to trigger an application on arrival of the last message in a queue?

- Should (can) the sending application create a trigger message after the put of the last "data" - message?
- Any way the triggercontrol attribute could be used (queue attributes)
- Any other ideas?

(I hate synchronizing an asynchronous transfer)

**Answer**

You could define your Q with say TRIGMPRI(9) & TRIGTYPE(FIRST). Put your msgs with a lower priority & put the final msg with Priority 9.

or...

You could put the "last" message to a DIFFERENT queue and trigger on that one.

or...

One way I have seen is as follows.
Sending appl sends a message to receiver "start of transfer" Receiver is triggered, and creates a permanent dynamic queue, whose name it sends back to the sending appl.

Sender now sends data messages to the permanent dynamic queue, which is not triggered. After sending the last message, send a second message to the receiver "end of transfer", naming the permanent dynamic queue.

Receiver is triggered again, reads and finally deletes the permanent dynamic queue.

or...

What about using two queues, a data queue and a control queue. The supplier of messages sends all its messages to the data queue and then either puts the last segment of the message or a separate control message to the control queue. Triggering would then be easy, triggering by any condition on the trigger queue would only happen when the set of messages are all available on the data queue.

or...

Since a queue can be triggered based upon priority, send all messages as priority less than 9 except the last. Have the priority for triggering set to 9. No trigger will happen until you send a priority 9 message. Once the last message is on the queue (priority that satisfies trigger conditions), the trigger will occur and you can then process the other messages on the queue. This is a nice way to process a batch of detail records and then put your "batch header" out as a priority 9 with a count of how many "detail" messages there are as well as, perhaps, a batch total.

or...

Any mechanism that tries to use priorities can disturb the sequential delivery of the messages that you also need. For example, if the network is down then the messages will wait on the transmission queue. When the network restarts the higher priority 'last' message will get sent before the others so will not show that the others have arrived.

Alan Powell

Use of trigger queues, initiation messages and CSQQTRMN

Question

We've got our OS/2 MQ server inserting messages into a host trigger queue which then inserts an initiation message into it's initiation queue and, via CSQQTRMN, starts up an IMS transaction.

Our question is what is the recommended triggering and IMS transaction structure for a large on-line response type mode application?

We started with the trigger queue triggered every and our IMS transaction simply doing a get unique, reading one message off of the trigger queue, processing it, and then doing another get unique (a standard response mode type format for an IMS transaction). Of course, if the IMS transaction abends, or if the trigger queue is get inhibited, etc., no initiation message left in the initiated queue, so the last message in the trigger queue is never processed until another record is inserted into the trigger queue. Is there any way to ensure that the number of messages in the initiation queue is the same as the number of messages in the trigger queue?

The other way to do it is just to have it triggered first with a trigger interval, or triggered depth, and have the IMS transaction do a get unique, and then goes into a loop that reads all of the messages off of the queue in turn,
performing an IMS checkpoint after the processing of each message. Then performing a get unique after all of the messages are read.

We would like to use the trigger every solution (we are a large shop and would like to keep the processing of one message as one IMS transaction as the stats regarding the number of transactions, average response time, etc. Are important to us, for billing, tuning, etc.) however we can't see a way to always ensure that every message in the trigger queue will be processed.

Do you have any suggestions?

**Answer**

The design of triggering suggests that a triggered application should read all messages off the application queue.

The MQM APG, under heading Trigger Messages without Trigger Events describe how MQM writes extra trigger messages to the initiation queue.

When a trigger monitor opens an initiation queue, if the application queue has at least one (trigger type FIRST or EVERY) or trigger depth (trigger type DEPTH) messages on the queue, and is currently not being served (i.e. is not open for input), the queue manager will write a trigger message to the initiation queue (assuming other trigger attributes, as defined in the APG, are correctly set).

Also note, when the last program serving an application queue defined for trigger type FIRST or DEPTH, closes the queue, the queue manager will write a trigger message to the initiation queue if there is at least one (trigger type FIRST or DEPTH) messages on the application queue (assuming other trigger attributes, as defined in the APG, are correctly set).

In your triggered application, note that as documented (in Application Programming Guide, under Syncpoints in IMS applications) when you call GU (this counts as a syncpoint) the MQM adapter closes your queues and disconnects you. This action will give rise to the trigger messages from queue closing/

Also, at restart time, if there are messages on a queue, MQM will write only one trigger message for a queue with trigger type EVERY even if there are many messages on the queue.

Another point is that if the trigger monitor should end (normally or otherwise) then no trigger messages will be put on the initiation queue for messages subsequently written to the application queue.

I think the upshot is that if your triggered application does not read the application queue to exhaustion at each invocation, then some messages will never be read.

---

**MQ/NT C++ Compiler**

**Question**

Does MQ/NT version 2 supports Microsoft Visual C++ version 4.1 or later? By specification, it is up to version 2 only.
Appendix A of the Windows NT System Management Guide says:

Minimum supported levels are shown. Later levels, if any, will be supported unless otherwise stated.

- C, using MS Visual C++ V2.0

So, the original testing was done with V2.0, versions 4 and 5 were verified when they were released.

**OS/2 queue manager access**

**Question**

I am writing an OS/2 MQSeries application using Visual Age C++ V3.0 as the compiler. This application is written in ANSI C.

I have successfully compiled and linked the portion of the application which does the MQSeries calls into a DLL. I have successfully compiled and linked a driver program to call the DLL. When I try to run the program I get an exception (XCPT_ACCESS_VIOLATION) on the MQCONN instruction.

The queue manager is defined and started. I can compile, link and run the sample programs (one, at least; that is all I tried to make sure the rest of my setup was correct).

Has anyone else seen behavior like this? Could any provide an explanation and possible changes to allow this to work?

**Answer**

Did you specify the correct /Ge flag whilst compiling the modules that you linked together in the DLL? Perhaps you could append the options and flags you used in producing your code if that doesn't help.

**Response**

Perhaps not. The compiler lines from the make file for the DLL follow:

```
c.obj:    
@echo " Compile::C++ Compiler "   
icc.exe /Tdc /Sa /Q /Ti /Rn /Ge- /G4 /Fo"%|dpf.obj" %s
```

**Answer**

/Rn generates code that can be used as a subsystem without a runtime environment.

Are you sure you meant to specify that flag? Try /Re (which is the default if omitted).
Response

I am trying to set this up to run on a system without the runtime. I could package the runtime with the module; I was just trying to avoid carry the extra component separately.

I tried changing /Rn to /Re. I receive the same error. (I made the change in both the DLL makefile and the EXE makefile).

Answer

Please try the following (because I can successfully call MQCONN from a DLL):

Produce the DLL via:

\texttt{icc /C+ /Ge- /Gd+ /Gm+ /Su /Ti+ (for each of your dll objects)}
\texttt{ilink /NOFREE /ALIGN:16 /EXEPACK /NOI /DEBUG /DLL <objs> mqm.lib}

Produce the main via:

\texttt{icc /C+ /Ge+ /Gd+ /Gm+ /Su /Ti+ (your main program)}
\texttt{ilink /NOFREE /ALIGN:16 /EXEPACK /NOI /DEBUG /EXEC <objs> mqm.lib}

If this still does not work, can you really strip down your source code to only call MQCONN in the dll, and only call this one function from the main function.

You could also try installing the latest Visual Age CSD.

(N.b. This was also tested on NT without any problems).
IMS

MQM 1.1.4 IMS Bridge: How can we prevent flooding IMS Q's?

Question

We are implementing several new applications which will use the IMS Bridge of MQM 1.1.4 to sent messages from MQ-managers (Tandem and also other MVS-applications). At least 2 of those new applications can put a lot of messages to the MQ-queues (via batch jobs). For MQ this is not a problem because you can isolate those queues. The trouble comes (we think) when the MQ-manager sends the messages (via OTMA) to IMS. Because the sending of the messages is much faster than the processing IMS MPP- application, we foresee that the IMS-queues will be flooded with MQ-messages. And from experience we know that this can kill the performance of the IMS-system (assumption: IMS queues are big enough). In one application about 500,000-700,000 messages are put on the MQ-queue in a few minutes.

We are looking for solutions how we can reduce the amount of messages from MQ to IMS. Are there options to slow down the traffic and what are the consequences of those options for other applications or MQ-IMS resources?

If anyone has some solutions, please tell me. Thanks in advance.

Answer

There are no options in MQ to slow down message delivery; in fact there is very little demand for slowed delivery, we get many requests for faster delivery.

I think that the solution will have to be in the application design (of either the feeder application or the IMS transaction) or the IMS system setup.

The message rate you mention didn't seem excessive for an IMS system to handle. Have you actually put 500,000 messages into MQ in a few minutes? This is a higher throughput than our design target. I know we have achieved sustained rates of >13,000 messages put, transferred, processed in IMS, returned, got by application per minute during testing (with little IMS workload in the fastpath transaction).

Response

Let me explain a bit more:

At this moment we don't have experiences with such a load. We are now in the phase of the technical design for a new Domestic Payment system and we are afraid that we may impact our IMS performance. One of the feeding applications is an old (legacy) batch application for periodic transfers. So (at the first day of the month) this application will 'generate' 500,000-700,000 messages in a batch job, and send them to IMS. Whether this will be in 5 minutes or perhaps more, I don't know yet (we didn't do any tests so far). As I already said, we are just afraid that this CAN happen. We estimated that the IMS MPP will have an elapsed time of 0.2 seconds, and runs limited parallel to avoid database locking problems. The problem depends of course on the fact how many MQ messages per second will be send to IMS (via the bridge).
I had hoped that there were some options to specify perhaps the number of threads per MQ queue or other things to slow down the transfer rate for a specific MQ queue, but I read from your answer that the feeding application must do this (IMS itself cannot help us).

Is it possible to send messages/transactions by the bridge synchronously and wait for the output before sending a new message? (A kind of one way traffic or half duplex)

Is it perhaps a solution to define the IMS transaction as an IMS fastpath transaction? In a 'normal' terminal environment you have a dedicated buffer/queue instead of the general IMS queues. Is the MQ bridge one (1. 'terminal' for IMS? Can we use this 'bypass' to slowdown the input of messages?

I wonder whether we are the first who foresee this potential problem.

**Answer**

You are not the first to raise the matter, a colleague of mine provided the attached answers to questions originating from Spain I've attached the full text of the questions and answers, I hope it is useful.

**Question**

If a customer decides to use MQSeries-IMS bridge in a MVS/ESA only environment, with no data sent or received from other environments, does he need to code data-conversion routines for any kind of data? (i.e binary, packed and floating point).

He is worried about this because the MQSeries Application Program Guide manual states in the 'Writing MQSeries-IMS bridge applications' topic that it's necessary when coding for MQSeries bridge. Although, we think we have misread any think because all data is EBCDIC.

**Answer**

You are correct. If done entirely from within a mainframe, no data conversion whatsoever is needed. You will need to indicate whether the message contains an IIH prefix or not (MQMDFormat = MQIMS if IIH present; with any other format name the bridge assumes no IIH).

**Question**

With the MQSeries-IMS bridge, is there any way of preventing an over-run of the IMS queue? Is it possible to stop MQSeries from putting more messages.

**Answer**

There are several ways this could be done. I have detailed two below:

The user first needs to be sure that it is MQ which is filling up his queues, and not legacy terminals.

Assuming it is MQ, he could periodically monitor the queue depth, using IMS command
and if the total number of messages queued is excessive, issue an IMS command

    /STO TMEMBER xxxxxxxx TPIPE ALL
    or /STO OTMA
    or issue an MQ command to GET-DISABLE the bridge queue(s)

until the queue depth is below a user defined threshold, then issue

    /STA OTMA
    or /STA TMEMBER xxxxxxxx TPIPE ALL
    or issue an MQ command to GET-ENABLE the bridge queue(s).

Alternatively, he could have an IMS Automated Operator task looking for the message:

    DFS2015 NUMBER OF RECORDS IN LMSGQ DATASET HAS EXCEEDED UPPER THRESHOLD

This message is issued by IMS when the queue is about 70% full.

Then issue /STO TMEMBER.... or /STO OTMA or MQ Get-disable as above.

Then wait until the following message appears in IMS:

    DFS2018 NUMBER OF RECORDS IN LMSGQ DATASET IS NOW BELOW LOWER THRESHOLD

this appears when the queue depth falls below 50% full.

Then issue /STA OTMA or /STA TMEMBER...... as above.

I would prefer the approach where the DFS2015/DFS2018 messages are monitored. If it is easy to disable bridge queues, I would prefer that to stopping TMEMBERs or OTMA in IMS, because the reply messages could continue to be sent from IMS to MQ, further reducing the pressure on IMS queues.

Another way of reducing pressure on queues is to make the messages Send-then-commit (ie Commit mode 1). Although such input messages are still queued, output messages are not. This increases region occupancy, however.

MQSeries IMS-Bridge with messages > 32K

Question

I'm writing a program that puts messages on a MQSeries Queue to OTMA. MQ is able to handle messages up to 4MB.

1. How must a transaction in the IMS be written in order to handle messages of that size?

The MQ guide says to use the following format for segmented messages:

    <MQIIH>LLZZ<transcode> <data>LLZZ<data>LLZZ<data>....

MQIIH is the header for OTMA. Each LLZZ<data> segment must be < 32K.

I've been told that even IMS segmented messages are limited to 32K.
Discussion

This is very interesting to me because I have already explored all these things that you are now exploring, and I found out some very interesting facts. We also are trying to send messages from outside of IMS (from the internet, actually), to an IMS application on MVS/ESA.

In theory, IMS will support multiple segments of 32k each, and I don't know the IMS limit, but I know that all segments must exist in the same physical message. Thus, if the message is coming from MQSeries, then 4M is your limit.

However, in practice, even 4M is too much for the IMS queues to handle, and it depends on how the IMS regions and queues are defined. In my IMS system, for example, I was lucky to get a 1M message in the queue, but I wouldn't count on it in production, and even if it were true, multiple such messages would definitely not be allowed. So I think IMS is your real physical limitation, and that is why I chose not to use OTMA as the interface between MQSeries and IMS.

What we are doing, therefore, is using the IMS trigger monitor on an initiation (trigger) queue which is triggered when a specified application queue receives data (possibly even > 4M) from the internet (ie another distributed queue manager). CSQQTRMN is the predefined trigger monitor for MQSeries for IMS, and it will simply start my IMS transaction with a short trigger message. Then my IMS application will read the MQ application queue and store the data in a buffer. Interesting, but I can allocate 8M easily in a PLI program running in IMS, but the IMS queues cannot handle this.

To support messages > 4M, my application program in IMS will simply put all the 'user-segmented' 4M messages together into one long message by reading the MQ application queue several times.

For bulk data between the outside world and IMS/ESA, I think this is the best solution we've been able to derive. However, if anyone knows of a better one that is more practical (and works), then I'm certainly open to alternatives.

Response

First of all: Thank you very much, that was more than I expected.

In fact, I don't need messages bigger than 4M yet, but only to understand your append completely the following:

You trigger an IMS transaction that reads the MQ Application queue. And if you have a message bigger than 4 MB, you divide your message into several 4 MB MQ messages and your IMS transaction puts them together again. Is this right?

Maybe the following is only because of a lack in IMS knowledge:

You wrote, that you allocate 8 MB in your program, but IMS cannot handle this. If you make it like the above, what do you need for the IMS queue?

Discussion

On the first question, you are correct, in my IMS application I read the MQ application queue and put all the 4m segments together in a buffer.

On the second question the IMS message queues cannot handle large messages, but an IMS program CAN allocate much more data. This has to do with the way that IMS is defined. It seems illogical, but IMS allocates and
maintains message queue storage, but PLI allocates and maintains program data storage from a different pool (I don't know the particulars).

**Answer**

IMS can handle large amounts of data. During test of the MQ-IMS bridge, we have successfully put 4MB messages from MQ onto the IMS message queue using OTMA. We had to tune our small development system to make this work.

Your problems with large messages sound similar to ours:

- development system - small by design
- region size too small
- Long Message Dataset too small
- QBUFs needed increasing
- etc.

With those sorted out we successfully handled the data from a 4MB MQ message through the MQ-IMS bridge - both into and out of IMS.

A solution using the adapter and CSQQTRMN is equally valid, possibly better; BUT don't stay under the misconception that either IMS or the MQ-IMS bridge is unable to handle 4MB.

Specific answers to the initial questions are:

1. 4MB is the maximum MQ message size
   - 32k (-4 I think) bytes is the maximum single segment size in IMS
   - BUT an IMS transaction can handle multiple segments if it is designed and genned to do so.

2. The source who said: "that even IMS segmented messages are limited to 32K" was either incorrect or misunderstood; he may have been trying to say "that IMS message segments are limited to 32K"

3. which is correct.

**Response**

I accept the fact that you have proven that the IMS message queues can handle 4M messages. But I must ask the following:

1. Can a normal IMS shop assume this as well? Implication here is did you have to change IMS parameters significantly enough such that a normal shop would not or could not do it?

2. With how many of these 4M messages did you flood your queue? The implication here is will the IMS message queue accept only one of these, tens, or even hundreds of these without problems?

I think it's great what you have done, and I only want to ensure that it's something that all or most production IMS shops could handle.
Answer

You're going to have to work hard to get all the parms fixed to make the message queuing system (normal size 4K'ish) handle 4M messages. That's an order of magnitude bigger.

Since DEVELOPMENT IMS systems are not usually set up to hold many messages, they could have difficulty holding even one 4MB message. Talk to your production people about what changes would be needed to support your 4MB messages in the production system.

Response

Since I've already proven that my IMS system cannot "normally" handle large messages, I am using the IMS trigger monitor and am letting the IMS application program handle the obtaining of storage for the MQGETs via the IMS Adapter.

Answer

The solution you are proposing is at least as good as using the bridge.

Accessing MQ IMS-Bridge from MQSeries Java

Question

Is there a way to access the MQ IMS Bridge directly from a MQJava applet?

I read in the MQ for MVS/ESA System Management Guide that the application program and the IMS adapter are running in the same address space. Does this mean that we need to go throw an MVS program that gets the msg from a queue and puts it back to the IMS queue using MQFMT_IMS?

Answer

I'm sure you can access the MQ IMS bridge from a Java applet; all you need is to MQPUT an MQ message of the correct format, get it to MVS and onto a bridge queue.

The MQ IMS bridge is a separate feature from the IMS adapter. The IMS adapter runs in the address space of the IMS program (transaction) processing the message; this does not apply to a program originating a message to be processed by the IMS adapter.

There is no need for an intermediate program of the type you suggest.

IMS BMP simultaneously connected to multiple queue managers
Question

The books describe that for IMS, only queue managers which are listed in the subsystem definition table (CSQQDEFF), and listed in the SSM table in IMS, are connectable. However, the books seem to leave it open whether an IMS BMP can be simultaneously connected to two different queue managers, or whether the BMP must disconnect from the first queue manager before it can connect to the second queue manager.

Specifically, we wonder whether MQSeries/MVS will produce a 2103 return code (MQRC_ANOTHER_Q_MGR_CONNECTED) when a BMP tries to connect to a second queue manager, or whether the 2103 return code is applicable only to certain other, non-IMS MQSeries operating environments.

It would also be helpful if we would understand what the connection handle that an IMS BMP receives back from an MQCONN call really is, and who generates that handle:

- is the handle generated by the queue manager or by the IMS adapter?
- is the handle a pointer to a shared memory area that is used to pass data back and forth between the BMP and the queue manager? If not, what is the handle then?

Answer

Like the MVS/TSO adapter, the IMS adapter allows simultaneous connection to multiple queue manager instances. The connection handle, hconn, is generated by and has meaning to the individual adapters but not to MQ in the MQ address space.

The hconn is the address of a control block and the type of data in the control block is different for each of the three adapters.

IMS Bridge and CommitMode

Question

Can you clarify the impact of using the different CommitMode MQICM_COMMIT_THEN_SEND (0) or MQICM_SEND_THEN_COMMIT (1)? For example, is there a significant difference in performance? Does synchronous also imply serialized if multiple messages are put to the same queue that the IMS bridge will service? Is the once and only once message delivery not supported when using commitmode 1?

Answer

CommitMode 0 enables recoverable, once and only delivery, CommitMode 1 does not - there is some overhead in achieving this! Any difference is less than the effect of running your MQ or IMS system DASD unoptimised, and certainly much less than the difference between persistent and non persistent MQ messages. You should benchmark your proposed configuration.
**Question**

Does the IMS bridge follow the standard message ID and correlation ID model for request/reply messages? By that I mean copying the message ID of the request message to the correlation ID of the reply message.

**Answer**

Yes, no second message will be sent until IMS has acknowledged receipt of the first; NOTE: as soon as IMS has accepted the message the next can be sent, there is no requirement that the message is processed in IMS before the next is sent.

**Question**

Are there other hints/tips that one should be aware of when using the IMS bridge?

**Answer**

Once and once only message delivery is only achievable using commitmode 0.

---

**IMS BRIDGE: Local or MONOPLEX MVS System**

**Question**

Concerning the IMS bridge, should the MVS system be a LOCAL system (which means no couple data sets) or a MONOPLEX? Are there any functional differences for MQSeries between a LOCAL and a MONOPLEX MVS image?

**Answer**

The MVS definition can be either LOCAL or MONOPLEX in the case you're asking about; the XCF macros used in the code do not use couple datasets.

There is no MQ functional difference; the XCF differences between MONOPLEX and LOCAL do not affect the MQ function.

The MQ IMS bridge does not pre-req the MQ IMS adapter - they are completely separate.

There is no MQ maintenance required. You must have at least PN84198 on your IMS system; there is other OTMA-related IMS maintenance which you should install but OTMA is not enabled without PN84198.

---

**Installing the IMS Bridge**
Question

If I have no requirement to make MQ calls from an IMS application, but do intend to use the IMS Bridge, I believe I do NOT need to perform the customisation steps to install the IMS Adapter, only the steps to install the IMS Bridge. Is this correct?

Answer

Yes, that is correct.

MQ and OTMA Manuals

Question

Where I can read more about MQ and OTMA?

Answer

List of manuals and references to read.

IMS

- IMS/ESA V5 Open Transaction Manager Access User's Guide, SC26-8026-01 (the -01 is important)
- IMS/ESA Application Programming: Transaction Manager, SC26-8017
- IMS/ESA Customization Guide, SC26-8020

MQSeries

- MQSeries for MVS/ESA System Management Guide (SMG), SC33-0806-04
- MQSeries App.I Programming Guide (APG), SC33-0807-06
- MQSeries App.I Programming Reference (APR), SC33-1673-02

And where within them, topic by topic

- The MQSeries-IMS Bridge - SMG. ch 3.5, p.176
- Customizing the MQSeries-IMS Bridge - SMG. ch 2.3, p.90
- Controlling the MQSeries-IMS Bridge - SMG. ch 3.6, p.188
- Using OTMA exits in IMS - SMG. Appendix B, p.423
- Writing MQSeries-IMS bridge applications - APG. ch 15, p.181
- MQIIH structure and constants - APR. ch 2 and 6
- Mapping MQ messages to IMS transactions - SMG. Appendix C, p427 (will move to APG)
- Resetting tpipes - SMG. Appendix C, p428 (will move to Command Reference)
- Messages and codes
– IMS Messages and Codes
– MQ for MVS Messages and Codes
– SMG. Appendix C, p430 (will move to Messages and Codes)
MQ 2.0 for NT - Domain user vs. local user

Question

I am trying to install MQ 2.0 on a NT server running 4.0 (a server in a domain). My user id is an domain administrator. I can install MQ and run crtmqm if I sign on to my local machine. But if I sign on to the domain, then I can not configure MQ at all. The System Management Guide does not seem to provide an answer on how to configure and manage the MQ manager when sign on to the domain.

The server needs to be signed on to the domain all of the time. Does any one have a way to configure and manage MQ when signed on to a domain?

Answer

This is easily fixed. There are a couple of ways of doing this. Firstly, on the workstation where you are running MQSeries, ensure that the Domain Administrators group is a member of the local administrators group. You should not be logging on as "Administrator" - this is a 13 character Userid, and MQSeries will not recognize it.

The other way is to create a group called 'mqm' on the Domain SAM and add your userid to that new group - that should solve the problem.

Authorization to put reports on queue using MQSeries for NT v5

Question

When messages bound for an MVS system over an LU 6.2 channel expire and I have set the MQRO_EXPIRATION flag in the report field, I get a message in the NT event log that says the report message could not be put on the reply-to queue. The reason code is 2035 (MQRC_NOT_AUTHORIZED). The queue and queue manager names are correct.

Using dspmqaut -m <q-mgr> -t q -n <q-name> -g mqm I get:

Entity mqm has the following authorizations for object <q-name>: get browse put inq set dlt chg dsp passid passall setid setall clr

When I receive messages from the application running on MVS they get put on the queue with out any problems.

What group/userid does the program that puts expiration reports on the queue run as? Is there some other authorization that I need to check?
Answer

When exactly do the messages expire? Have they just been received from the queue manager on MVS, and are being put by the receiving channel. If so, check the MCAUSER and PUTCTX fields of the channel definition.

Installation of Windows NT Version

Question

I had installed the Windows NT version of MQSeries several times before without any problem. However, when I tried to install it on a customer's system (NT 4.0) I ran into some problems.

- First of all, after installing the product under the Administrator userid, I noticed that the 'mqm' group wasn't created automatically.

- When I invoked 'crtmqm /q SAMPLE' under the 'Administrator' userid, it returned without any message and without creating the queue manager. I expected it to complain about 'Administrator' being > 12 chars long.

- When I invoked 'crtmqm' under an 'mqmadmin' userid, I got the message "You are not authorized to perform the requested operation". I got the same message after adding 'mqmadmin' to a manually created 'mqm' group, even after I reinstalled MQSeries (with 'mqm' group pre-defined).

- I had also tried installing the product directly from 'mqmadmin'. In this case, I got no message when I invoked 'crtmqm' from this userid.

- Errors were recorded in \mqm\errors\amq0023.0.FDC file. Does the "WinNT error 1332 from LookupAccountName" indicate a problem with the system?

Any suggestion and help will be greatly appreciated.

Answer

The problems you've hit are due to the installation failing to complete correctly. The installation attempted to create a local mqm group, but this failed - most likely because the machine you're installing on is configured as a backup domain controller. After this failure, the registry entries were not created as they should have been.

To avoid the problem in the first place, create a local mqm group on the primary domain controller before you install on a backup domain controller.

To recover after the problem, un-install MQSeries, create a local mqm group on the primary domain controller, and re-install on the backup domain controller.

FYI - the MQSeries for NT installation CD was refreshed in Dec '96 to avoid the 'installation on a backup domain controller failing to complete correctly if no local mqm group found' problem. The readme.txt of the refreshed CD is dated Dec 9th 1996. You will still need to manually create a local mqm group on the primary domain controller before you can actually use MQSeries.
MQ NT Qmgr Disconnect on login

Question

On our NT 4.0 server when Qmgr is running and I select "close all programs and log in as another user” the queue manager shuts down and has to be restarted manually.

In the NT control panel services MQ is selected to begin at startup.

My question is. How can we keep Qmgr running all the time when we have a need for different people to log onto the server for things such as backup, SQL admin etc.

Answer

When you install MQSeries for Windows NT, a service entry is added called IBM MQSeries. It does not do anything until you use the scmmqm command (described in the System Management Guide) to tell it exactly which MQSeries features you want autostarted.

Question

For some reason when running runmqchi from either a command line or from a command file designated by scmmqm it hangs the dos session and I have to ctrl-C out. strmqm works fine.

This happens when all defaults are accepted or when we explicitly state them on the command line . It does reject invalid arguments. The channel and listener can be started using runmqsc commands without apparent problem.

Any thoughts on what the problem could be?

But, assuming we can get the startup command file to work does this still mean that our NT server cannot be left in the "please log on” state and still have Q manager running like SQL Server and IIS do?

Answer

When you say that the channel initiator hangs the DOS session, how do you know this? The only text that the initiator writes to stdout is something like: 84H2004,6539-B43 (C) Copyright IBM Corp. 1994, 1997. ALL RIGHTS RESERVED

It is now up and running, although it will not return control to your DOS Prompt until you end it (using Ctrl-C). Use "start runmqchi” if you want to start the initiator in another window.

The purpose of scmmqm is to cause MQSeries to start when NT is booted. This means that MQ can send and receive messages even though a user hasn't logged on. It also means that MQSeries does not shut down when a user logs off.

As a bit of background: The MQSeries commands have a number of of prefixes the most common are:

str (eg strmqm, strmqcsv)

These commands submit a background process to do the work and return control.
run (eg runmqchi, runmqslr, runmqsc)

These commands do the work synchronously, ie. the command does not return until the process is complete.

As a consequence if you want to run a 'run' type command from a command window and do not want the window tied up until the completion of the command you must issue it as a background command. eg. detach runmqchi (OS2) or runmqchi & (unix)

MQ/NT & Multithreading questions

Question

Is it possible to connect under one thread and issue PUTs under a different thread in the same process? I saw some previous references to an OS/2 (I believe) situation, where it was stated that each thread had to connect before it could write to a queue. Is that true under Windows NT? The program I've written (in which one thread connects, and multiple others PUT to the queue) receives an "Invalid Connection Handle" error, although I'm sure the handle is correct when the PUT is issued.

Answer

Both the MQI Technical Reference and the Application Programming Reference define the scope of the handles returned by MQCONN and MQOPEN as follows:

The scope of the handle is restricted to the smallest unit of parallel processing in the environment concerned; the handle is not valid outside the unit of parallel processing from which the MQCONN or MQOPEN call was issued.

(See the descriptions of the "Hconn" and "Hobj" parameters of the MQCONN and MQOPEN calls, respectively.) So, for Windows NT the scope of an MQ handle is a thread.

Question

Another basic question: Person A logged on to NT and installed MQ. They are able to start MQ fine, and the samples work. When I log on and try to start MQ, The Qmgr starts fine, the Channel process starts and immediately dies (as does the Listener process), indicating that it couldn't connect to the Queue Mgr. I'm sure I'm missing something simple, such as NT permissions or some such thing, but we couldn't find anything.

Answer

Make sure that the channel definitions are correct and correspond to the Qmgrs defined. Make sure that your userid is in the group "mqm" as well as be defined locally.

MQSERIES for WIN/NT Setup Failure.
**Question**

Investigating the problem of installation failure on WIN/NT 4.0, I suspect that the trouble is due to the server configuration that is a Backup Domain Controller; in this case, as various Apars mention, to successfully install the problem I need to add manually an MQM Group on the Primary Domain Controller and resynchronize the domain before reinitiate the setup on my server. The question is: how can I manually create the MQM group? Unfortunately we cannot install MQseries on Primary domain controller, so the manual definition of MQM group is mandatory.

**Answer**

Start->Programs->Administrative Tools (Common)->User Manager for Domains

From within this application, do:

User->New Local Group

**Response**

Thanks, setup with manual definition of MQM Group was successfully.

---

**Permission problem with MQSeries for Windows NT**

**Question**

Using MQSeries for NT. They installed the product but when they try to continue creating the Queue Manager they get the error message:

'MQSeries was unable to display an error message'

What would be the corresponding recommendations regarding MQ for NT?

**Answer**

If MQSeries for Windows NT displays the message -

'MQSeries was unable to display an error message'

it is possible that the MQSeries messages file cannot be found.

The file is located by reference to this registry entry:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\----> -->\EventLog\Application\MQSeries\EventMessageFile

The file name is AMQCATNL.DLL. The registry entry should include the full path, eg -

F:\MQM\BIN\AMQCATNL.DLL
Both the file and the registry entry should have been created during installation of MQSeries.

### MQSeries for Windows NT on a Backup Domain Controller

#### Tips

When installing MQSeries on a Windows NT Server that is also a Backup Domain Controller (BDC), the install function is not able to create the MQM group. In addition the MQS.INI is not properly created.

If you create the group first, you will still get "Unexpected error, 2195". What the installation then is missing is the creation of the MQS.INI file and therefore you are not able to create a Qmgr.

What you will have to do is to either copy an existing MQS.INI file from another Qmgr (the file is common to all installation) or you will have to dummy-install MQ on your PDC (primary Domain Controller) and then take a copy of the MQS.INI file. Yet another way is to create a MQS.INI file with the entries described in the System Management Guide.

The file should be in the rootpath:\MQM directory.

In the second case you just have to install it (not create a Qmgr), copy the file and the uninstall it. It takes approximately between 10 - 20 minutes.

### MQSeries for Windows NT and Local/Domain logon problem

#### Question

MQSeries is installed on a NT server which is running in a domain environment.

If local logon is done we receive error 1332 no MQM function stack available. This error can circumvented when putting the mqm group into the domain. The local user does not have to be in this group. Only this group has to be existent.

What is happening during logon to the MQSeries instance concerning the NT security. Why must the mqm group be available in the domain even for local usage?

#### Answer

When MQSeries runs on NT, it may create objects that have restricted security access permissions. These objects may be accessed either by administrators or by members of the 'mqm' group.

MQSeries uses the context of the userid that is executing at the time. If the userid is logged onto the domain, MQSeries will look on the domain controller for the mqm group. If the userid is logged on locally, it will look for the mqm group on the local machine.
MQ Series Backup Failure

Question

I am experiencing a problem executing a system backup. I am running MQ series, Version 2.001 with CSD U200059, on an OS/2 Platform Version 3.0 with CSD 3003. I followed the backup instructions in the MQ System Management Guide. I shut down MQ using endmqm, then, after backup, I execute restart using strmqm. MQ series does not start up however. I receive the following messages:

Queue manager cannot be restarted because processes that were previously connected are still running.
AMQ8042 Process 75 is still running
AMQ7018 The Queue Manager has stopped

Note: Process 75 is runmqlsr.

This same cmd file runs without errors on a test machine but fails on the production machine. I might also note that it will run successfully on the production machine shortly after reboot but fails several hours later. I do not have any idea what the exact wait time is nor do I know what precipitates the failure. Flowmark is also running on this machine.

Has anyone seen this behavior? I would appreciate any suggestions about how to determine why runmqlsr is not shutting down and how I might resolve it.

Answer

Ending the queue manager does not terminate any listeners. This is so that if a remote QM tries to start a channel to the shutdown queue manager, the listener can tell it that the queue manager is unavailable, rather than the remote QM just getting a TCP/IP failure.

The V5.0 release of MQ has a new command called endmqlsr which will end any listeners that are running when the QM has ended.

Authorization for OS/2

Question

When using RUNMQSC under OS/2 to issue commands against MQSeries MVS/ESA 113, an authorization error occurs. RACF for some reason is expecting a userid called OS/2. Is this needed? Where is this documented? I would presume that this is an attempt to impose some sort of security over OS/2 remote access?
Answer

Because there is no security-manager on OS/2, MQI puts the standard-string "OS/2" in the userid, when the Put-Message-Options for CONTEXT were not set by program. In this cases the program can put the userid by himself in the message-descriptor. More about this can be read in the manual APPLICATION PROGRAMMING REFERENCE (Structure MQMD, Field UserIdentifier).

OS/2 high depth and low depth settings

Question

I am look at MQSeries, and I am fairly new to it. I am confused as to exactly how the local queue attributes of qDepthHighLimit and qDepthLowLimit work.

Assuming that they are the default values of 80% and 20% and I enable both qDepthHighEvent and qDepthLowEvent, when the queue rises above 80% or drops below 20% the respective events are generated. (according to the doc). I could not get this to happen.

The questions I have are:

1. Are there other conditions that must be met before these events are generated?

2. What Queue does these messages go? I could not work out where they went, and how to control where they go.

The Doc put a lot of effort in explaining the trigger messages (and it is good), but it doesn't seems to say how Event messages work.

Answer

The best reference for high & low queue events is:

MQSeries Programmable System Management - SC33-1482-04, (in my version chapter 3.4)

You need to enable performance events at the queue manager (alter Qmgr PERFMEV(enabled)), enable the performance events you are interested in on the queue (high, low and full: QDPHIEV, QDPLOEV, QDPMAXEV), set the high and low thresholds (QDEPTHHI and QDEPTHLO), AND the maximum number of messages (MAXDEPTH).

You also need to have the performance event queue (SYSTEM.ADMIN.PERFM.EVENT) defined as a local or remote queue.

The high and low thresholds are a percentage of the MAXDEPTH attribute of the queue.
**MVS**

**Allocation of MQM/ESA LOG dataset, LOG archive.**

**Question**

We are preparing MQM environmental files for our production systems. We plan to use DUAL LOGGING and DUAL ARCHIVE and wonder if one has experience in calculating the optimum size for these datasets:

1. What is the recommended SPACE size (in term of 3390 cylinders or in Kilobytes) for one MQM LOG file?
   
   Is allocating the log file such that it will (exactly) fit on 1 cartridge a good method? Or is it better to allocate larger log files and try to postpone the archiving to a later period (after office hours).

2. Has the use of COMPACT=YES for IDRC devices much advantages?
   
   In the MQM SMG manual we read "specifying YES adversely affects performance" but we wonder what is the impact of COMPACT=YES.

Any suggestions are welcome, thanks.

**Opinion**

I think the same rules apply to MQM logs as to DB2 logs, i.e. If you are not using disk archiving+SMS migration then the best value is for around one tape worth of disk space be specified. The reason for not using IDRC is that the read backward command gets really slow and would usually be used in case of a rollback which is when you don't want to to be slow.

Consider RAID or DUAL COPY instead of software duplication - its much better.

**Question**

In our company we have the policy to use/implement data duplication by the software, if the software provides the possibility.

In case of MQSeries for MVS/ESA:

- for the LOG files we will use DUAL LOGGING (software) advantage: the BSDS will record both primary and secondary logs.
- for the pagesets we will duplicate the data using hardware (remote copy).
- for the meantime we will archive to cartridge, later we will archive to DASD.
Opinion

So are you saying the pagesets are remote recoverable but the logs are not? I must confess I can see no reliability differences between dual copy and hardware duplication (both after all are two copies on two disks). There is however palpably better performance from hardware copy, although of course by using software copy its only the logs themselves that are duplicated not the entire volume.

Still its good the site has such standards, so at the end of the day its your choice. I assume you have also tuned the layout of the logs to minimize contention.

Channel indoubt

Question

We are using V113. A LU6.2-channel between two MVS systems stopped and was indoubt. Where and how can I find the reason for this problem. Is the only way out here to resolve the channel? With commit or backout?

Answer

If you are going to restart the same channel, then you don't need to make any user intervention such as RESOLVE CHANNEL. When the channel restarts, it will re-synchronize the in-doubt automatically.

However, if you need to remove the messages from the sending system's transmission queue without restarting the channel or start another channel against this transmission queue, then you do need to issue RESOLVE CHANNEL.

To do this successfully, you need to determine whether the last batch of messages from the sending system have been received at the remote system. If you perform a DISPLAY CHSTATUS(<Channel name>) SAVED on both the sending and receiving systems for your channel, the last logical unit of work committed will be displayed. If these are the same, then you RESOLVE CHANNEL COMMIT, since the remote system has committed the last batch and they should therefore be removed from the transmission queue. In the case where the unit of work ids are different, a RESOLVE CHANNEL BACKOUT is necessary, so that the last batch can be re-sent.

As I stated previously, all this work is normally handled by the channel and need not concern you except slightly esoteric cases.

Channel initiator failure w/OS390 1.3

Question

I'm trying to configure MQSeries DQM w/o CICS and my channel init proc always fails. I see the messages that the initiator is starting using my parameters and then I get:
CSQX131I -MQ1 CSWXADPI 8 adapter subtasks started, 0 failed
CSQX015I -MQ1 CSQXSPRI 5 dispatchers started, 0 failed
CSV003I REQUESTED MODULE EDCXVNOT FOUND
CSV003I REQUESTED MODULE EDCXVNOT FOUND
IEA989I SLIP TRAP ID=X13E MATCHED. JOBNAME=CSQ1CHIN, ASID=003e.
CSQ3201E -MQ ABNORMAL EOT IN PROGRESS FOR USER= 625
CONNECTION-ID=CSQ1CHIN THREAD-XREF=
.. several more SLIP TRAPs and Abnormal EOTs
CSQX005E -MQ1 CSQXJST Channel initiator failed to start
CSQX010I -MQ1 CSQXJST Channel initiator stopped

Thinking that this might be a problem with C/370 vs. LE/370, I tried using the EDC.SEDCLINK (I couldn't find the xxx.SIBMLINK library specified in the MQSeries for MVS/ESA dod). This produced an abend 047 in CSQXJST before anything gets started.

Has anyone else seen this? Am I doing something wrong?

Answer

When changing from C/370 to LE/370 run-time, or in applying some maintenance, some of the installation jobs for distributed queuing will have to be rerun. See Chapter 7 of the Program Directory for MQ Series for MVS/ESA. EDCXV was renamed EDCZV in LE and resides in xxx.SCEERUN. I have it in PLPA via IEALPxx along with EDCZ24, CEEBINIT, CEEBLIBM, CEEPLPKA, CEEEV003, and CEEOLVD, and xxx.SCEERUN is in LINKLSTxx.

DMQ with and without CICS

Question

Can anyone share the pro and cons for DQM with CICS and DQM without CICS? What criteria should I use to choose which one to implement?

Answer

The stand-alone (S/A) message mover (MCA) is better integrated with MQ than the CICS mover. For example, channel events (start & stop) can be generated as event message from the S/A mover, but NOT the CICS mover. Also, S/A channels can be configured and controlled from the same ISPF panel as is used to configure other queue objects, as well as the ability to configure and control S/A channels thru messages to the command queue and from the MVS console. None of this is available with the CICS message mover.

I believe that published performance measurements show that the CICS message mover performs better than the S/A message mover.

I advise customer to go with the S/A message mover, unless the find the performance wanting. So far...all new customer's I've worked with are using the S/A message mover.

It is worth noting that many CICS/ESA customers are continuing to use the CICS MCA - even when getting MQM/ESA for the first time. However I'd suggest if you are planning to support large numbers of channels from
MQM/ESA e.g. To MQ workstations, then you think seriously about going to the non-CICS APPC MCA. This has System Management more integrated into the rest of MQ e.g. You can more easily write MQSC scripts to define large numbers of channels etc.. The non-CICS MCA also generates channels events e.g. When channels end, which you make want to integrate into your Network Management strategy at some point.

INTERTEST and MQSERIES on MVS

Question

Is anyone aware of any problem trying to get MQSERIES and INTERTEST to co-exist in an MVS development environment? If not, is there anything special that needs to be done?

We are brand new to MQSERIES and we're getting some indication that there might be a problem with the two together.

Answer

We don't use Intertest, but a similar product Xpediter. The only problem we encountered was the fact that the product asks whether you use DB2, and you have to answer YES if you are using MQM (because it uses the same subsystem interface I guess) in your application (and no DB2).

Persistent Messages MVS/ESA system.

Question

I am working on a project that is using MQSERIES on an MVS/ESA system which is talking to an RS/6000 Box. We have defined the Queues as persistent. The problem is that when we stop the MQ manager and bring the manager backup the Messages are gone. What do we need to do so we do not lose these messages? When we stop the MQ manager on the RS/6000 and then start it up again, the messages are still there.

Answer

We guarantee to deliver messages, only once.

Before you shut down the queue manager, please issue a Display Usage command for the page set that the queue uses, and check that the messages are indeed persistent.

The default persistence of your queue may be persistent, but the application can always Put them as non persistent.
Resolution

I did a Display Usage and found out that when our AIX machine was doing a MQPUT that is was putting the Messages as Non-Persistent. We changed the code and added MQPER_PERSISTENT on the MSGDESC. Now when we bring the Qmgr down and back up the messages are still there.

MQSeries ISPF panels

Question

I have set up the MQSeries ISPF panels. I cannot find documentation on the setup required to access information for a Qmanager other than the one I am connected to. Can anyone point me to what needs to be defined?

When I try to access another Qmanager I get message:
CSQ0018E Target queue manager name is invalid or unknown.

The target Qmanager is valid and up and running. I know I probably need remote queue definitions but I don't know the specifics of what tells the ISPF interface what remote queue to use.

Answer

Did you define a XMIT-Queue with the name of your remote Qmgr?

Response

Yes, I defined the XMIT queues and the ISPF panels work fine. I can now access queues, etc. on another queue manager.

MQSeries/ESA V1R2 timestamp used

Question

Does MQSeries/ESA V1R2, supports timestamp in UTC (Universal Time Coordination) mode. Meaning: When there is a time change (spring and fall), can we leave MQSeries running, knowing that the unit recovery timestamps, will always be reflected by the timestamps in the log. Or, do we need to bring down MQSeries and start it 1 hour later, for the time change.
Answer

According to the APR, Greenwich Mean Time (GMT) is used for the PutDate and PutTime fields, subject to the system clock being set accurate to GMT.

Basically, MQSeries uses the MVS/ESA STCK instruction to obtain the time.

Note that the former term, Greenwich Mean Time (GMT), is now obsolete and has been replaced with the more precise UTC.

MVS Listener

Question

One can display on the MVS console (DA panel on TSO) the fact that our channel initiator, our CICS region and our queue manager are running. Is there a way to know that the listener is running, without starting it from the MQ Panels?

Answer

You can use the DISPLAY DQM command. The response shows a series of messages showing the current status of the channel initiator, including whether the listener is started or not. For example:

```
/cpf DIS DQM
```

Open Overhead using DQM in MVS

Question

I am measuring overheads prior to application design and development. We have been told that OPEN is expensive. Our measurement indicates it is not significant in the MVS environment. Elapsed time to open a LOCAL queue has been measured at about 0.5 milliseconds and a close about the same. CPU utilization is not very significant.

Question. Is there any significant difference in opening a remote queue? If I understand it correctly the open of a remote queue is really just an open of the Transmission Queue to the remote queue manager. If this is true, then as far as the local QM is concerned the overhead would be about the same as a local queue open - Correct?

Also does any message/control information actually flow from the local QM to the destination QM at OPEN time? Does the remote QM open the destination queue at this time, or only when the message is transmitted?
Answer

1. You are right, there should be no significant difference between opening a local and a remote queue - for the reasons you give.

2. No information flows to the remote queue at open time.

3. On some platforms it is the connect that is the slow operation - see the various performance SupportPacs.

Reading the message descriptor from MVS

Question

I am working on a team that is using MQSeries to send messages back and forth on MVS and OS2. My problem is the messages I receive from MVS have a MsgId field in the message descriptor that appear to be garbage. My portion(OS/2) is written in C and the MVS sender is REXX. The rest of the message descriptor looks good. I have specified convert before my GET. The MsgId field looks like this: ÿÖûûæÜûû

We are new at MQSeries and would appreciate any help you can give.

Answer

This is working as designed.... The MSGID field does NOT get converted (neither does the correlation ID)

The MQI makes a distinction between fields declared as MQCHARnn, and those declared as MQBYTEenn:

- MQCHARnn fields are intended/assumed to contain printable characters, and so these fields in the MQMD are converted between character sets by the queue manager when necessary.

- MQBYTEenn fields are intended/assumed to contain binary or bit data, which is NOT converted (these fields do not contain printable characters)

Most string fields in the MQMD are MQCHARnn, but the "MsgId", "CorrelId", and "AccountingToken" fields are MQBYTEenn.

Decrease size of a pageset in MVS/ESA

Question

We want to decrease the size of a pageset. Is there a documented way to do that?
Answer

USERS OF THIS PROCEDURE SHOULD BE FULLY AWARE OF THE VARIOUS USES OF CSQUTIL.

1. System running with large pageset, most or all pageset is empty (ie: little or no message data on pageset).
2. The assumption is that the large pageset contains very little message data.
   This can be verified by checking the pageset usage via the DISPLAY USAGE PSID(n) command (see Command Reference for details).
3. Remove all users from MQM by altering security access. The only user that we want to have access to the system is the user MQ administrator using CSQUTIL functions.
   See the Systems Management Guide for details on security.
4. When all users are off the system run the COPY function of CSQUTIL to copy all message data from the large pageset. The COPY function is performed at a PAGESET level not at queue level.
   When this step is performed we will have a single sequential dataset containing all the messages that were on the large pageset.
5. Run the EMPTY function of CSQUTIL. This will remove all messages from the pageset. After running the empty function the pageset should be TOTALLY empty.
   This can be verified by checking the pageset usage via the DISPLAY USAGE PSID(n) command (see Command Reference for details).
6. Identify all storage classes that relate to the pageset that is to be reduced in size. Use the command DISPLAY STGCLASS(*) PSID(n). This may be performed using the COMMAND function of CSQUTIL.
7. Identify all queues that use any of the storage classes identified in step 6) via the command DISPLAY QUEUE(*) TYPE(QLOCAL) STGCLASS. This may be performed using the COMMAND function of CSQUTIL.
   Any queue identified needs to be altered to use a different storage class that maps to a different pageset. This DOES NOT have to be a permanent change to the queue, but is essential for the Qmgr to be able to restart. Failure to do this may result in 00C91B01 abends as the Qmgr attempts to restart.
   Alter the storage class attribute of a queue via ALTER QL(n) STGCLASS(stgclassname).
   This may be performed using the COMMAND function of CSQUTIL.
8. Stop the Qmgr via the STOP Qmgr command (quiesce or force).
9. Run the RESETPAGE function of CSQUTIL against all pagesets other than the pageset which is to be reduced in size.
10. Define a new smaller pageset to replace the large pageset. When steps 9) and 10) are complete we will have a completely new set of pagesets.
11. Define new log datasets (BSDS and active logs) with new dataset names
12. Restart MQM using the pagesets created in steps 9) and 10) and the new BSDS datasets created in step 11).
   Using these datasets means there is NO RECOVERY to be performed at restart time.
13. RUN the LOAD function of CSQUTIL to load back all the messages copied during the COPY function in step 4). The assumption is that there is a small amount of data to be reloaded.
14. Alter the storage class of any queues where the destination for messages is the new smaller pageset.
   This may be performed using the COMMAND function of CSQUTIL.
15. Reset security so that all users have access to MQM.
16. MQM should now be available for use with the smaller pageset.

## Dual Logging versus DASD Dual Write

### Question

In the MQM/ESA SMG Chapter 5.3 p252 there is a statement:

**Note:** You should always use dual logging and dual BSDS's rather than dual write to DASD.

Why? Surely hardware dual write performed by the DASD controller is transparent to MQ, and saves CPU cost versus software dual logging.

### Answer

This is an interesting topic.

There are subtle differences. When MQSeries for MVS/ESA does software dual logging, it first writes one copy, and when that operation has successfully completed, it writes the other copy. It does NOT start both I/O operations in parallel.

In certain hardware failure scenarios, primarily power loss during a DASD write operation, some DASD devices may damage the block being written. I think it may be because the CRC may not be written correctly as power is failing.

It's important not to damage the last block. While loss of the logical record being written during a power failure is to be expected, we have a problem when we are rewriting a physical block that contains (in earlier logical records) the COMMIT of a prior unit of work. The transaction ACID property of DURABILITY dictates that we not "forget" that we committed a transaction.

If the log tail contained an "UNDO" record, and we had updated a pageset, we might not be able to perform backout if we lost the last log block.

Most DASD mirroring hardware seems to stress throughput, and may well initiate both I/O operations in parallel, or with no synchronization to assure that both operations are not "at risk" at the same time. My friends who know about DASD controllers indicate that this is a model and implementation dependent decision, but in several models they're aware of there is no special care taken to assure that the operations to both copies are performed at different times.

In addition, when we do dual copy logging in software, we can react appropriately to the loss of ONE of the two copies.

If MQSeries for MVS/ESA observes a failure during log write it immediately switches to the next active log extent (to get back into dual-copy mode), and schedules an archive operation to minimize the period that we're dependent on the single copy.

If hardware mirroring is used, MQSeries for MVS/ESA is unaware of the 2nd copy, and can't do anything extraordinary when one of the two copies fails. I presume with hardware mirroring you have the choice of continuing along in single-copy mode (which is something we considered unsafe), or failing.
Dual software logging is not free. It primarily lengthens the response time of commit, rollback, and operations performed "out of syncpoint" to persistent messages (all of which require log "forces"). But it isn't as expensive as you might think.

Dual copies of the log are much more important than dual copies of the pagesets. We can perform media recovery for a lost pageset by traversing the log. We cannot recover completely if the log is lost.

## Handling of non-persistent messages in MVS

### Question

1. In our experience both persistent and non-persistent messages are written to the linear VSAM files. If we were to write a persistent message to a queue followed by say 20,000 non-persistent messages and then finally another persistent message and all are externalized to the linear VSAM from the buffers, when the Q. Manager is recycled, how does MQ maintain its message pointers? Is there an index, a hashing algorithm or does MQ read all persistent and logically deleted non-persistent messages to GET the second persistent message?

2. Do persistent messages share the same pages as non-persistent messages if they are in the same queue?

3. If we write a very large message, say 3.8 meg, does MQ ensure that this messages pages are all written from the buffer together or not.

### Answer

1. Use the largest buffer pools possible, and then the least number of messages will be put to disk. We purge non-persistent msgs at restart, and then first persistent message will point directly to the 2nd.

2. No. Separate sub queues exist for each priority and persistence in each queue. Pages are not shared by sub queues.

3. There is no need to write these pages together. Our Buffer Manager decides which stay in the buffer pool, and which get written to disk. Again, use the largest buffer pools for best performance.

### Background

Our patent number 5,452,430, "System and method for storing persistent and non-persistent queued data and for recovering the persistent data responsive to a system restart" describes the technique used in some detail. The patent is dated November 7, 1995.

Some of our design objectives for MQSeries for MVS/ESA were:

1. Non-persistent messages are the kind of thing you would normally think of as stored in virtual storage only. It's quick and relatively cheap, but you expect to lose them if the sytem is restarted.

2. Persistent messages are the kind of thing you think of as going to disk. You pay more, but you expect them to survive across queue manager restarts.

3. We wanted to have non-persistent messages "spill" to DASD if virtual storage was insufficient.

4. We wanted both persistent and non-persistent messages to benefit from caching, i.e., a buffer pool.

We log activity related to persistent messages so that they can be recovered at restart, whether or not the buffer manager has actually written the message to the queuing file (the "pageset"). If write has been deferred and the buffer is still in virtual storage at crash time, the log contains the information necessary to bring the queue file (pageset) up to date. This technique is common in databases (DB2 uses it), and is referred to as the "steal,"
no-force" buffer management policy. "Steal" means that we are permitted to write an uncommitted update to DASD (necessary for 4 meg MQPUTs with a small buffer pool), and the "no-force" policy means that we need not physically update the pageset at COMMIT time.

The "steal" policy implies that we must write UNDO information on the log, and the "no force" policy means that we must write REDO information on the log.

The "no-force" policy is great for message queuing systems when MQGET follows shortly after MQPUT. The message is written to a sequential log, but if sufficient buffer space is available the message doesn't need to ever be written to the pageset nor read from the pageset; all the necessary information is gleaned from the buffer pool. And if we crash there is enough on the log to update the pageset to its appropriate state.

The rule of thumb is that I/O to sequential files (like logs) is at least 100 times faster than random disk I/O. And as transfer rates are increasing faster than access rates, this number is going to increase in the future. (See Gray & Reuter, "Transaction Processing: Concepts and Techniques, page 53).

The net of this is that we can handle a much greater message rate as a result of the "no force" policy than we could if we were required to update the pageset at MQPUT or at MQCMIT time. This is done safely by use of traditional database techniques.

The best paper that describes the database techniques used is by C. Mohan, Bruce Lindsay, Don Haderle, Hamid Pirahesh and Peter Schwarz, titled "ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks using Write-Ahead Logging". The paper appears in Transactions on Data Systems 17(1), pages 94-162 (1992).

5. We didn't want anyone to have to choose how much virtual storage to allocate for non-persistent messages and how much to allocate to DASD cache for persistent messages. Instead, a single buffer pool uses straight LRU rules to keep the most active pages in the buffer pool, without regard to message persistence.

6. We didn't want to scan through non-persistent messages at restart time. Queue scans take a lot of time. Instead we put persistent messages in one linked list of pages and non-persistent messages in another linked list of pages.

   During normal MQGET processing we fetch the head of each of these linked lists (if both exist), and present messages in MQPUT timestamp sequence within priority. A real-time merge is done of the heads of the persistent and non-persistent "subqueues". Actually there are 10 priority levels so there may be as many as 20 "subqueues" or linked lists of pages per queue. Few queues ever really have 20 subqueues, so this isn't as complex as it sounds.

7. At restart we free the storage occupied by non-persistent messages by traversing all the "space map" pages which contain bit maps used for storage allocation. Pages containing non-persistent messages are simply freed at restart. Since one space map page contains bits for many thousands of data pages, this is a relatively quick operation.

   We initially thought of using the words "recoverable" and "non-recoverable", but this got very confusing when we also talked about messages being involved in units of recovery (commit scopes). Since both persistent and non-persistent messages could be part of a unit of work (or unit of recovery), calling a message "recoverable" got confusing. Separating the notion of whether a message is or is not going to survive restarts from the notion of whether a message operation participates in COMMIT or ROLLBACK is a difficult concept anyway. Calling the message "recoverable" would make the distinction even more difficult.

With the wisdom of hindsight I wish I'd used "volatile" and "non-volatile".

These techniques are used by MQSeries for MVS/ESA; other techniques are used by MQSeries on other platforms.

MQ APPC security
Question

I've been trying to work out from the books, if I set userid and password on a channel definition for an SNA channel into MVS, where in MVS is this userid and password defined? Oh, and is this what is called APPC Conversational security, or bind time security?

Answer

The user IDs and passwords are defined in RACF (or in whatever security product the installation is using).

Conversation security transmits userid and password on each conversation if you set SECACCPT=CONV in the VTAM definitions (SYS1.VTAMLST) or you can set CONVSEC(CONV) in the RACF definitions in the APPCLU profile. If both partners are on MVS systems, you could use CONVSEC(ALREADYV) which accepts just the userid without password.

Bind security (also called link security) happens at bind time. Both sides must have the same session key (aka bind password) defined. On MVS, session keys are defined in RACF in APPCLU class profiles. The actual verification is an exchange of encrypted random numbers. No user IDs are involved.

MQ Clients to MVS

Question

Does anyone have a summary of pros and cons for having a large number of MQ clients (Windows) connected directly to MVS as opposed to going via separate MQ server(s)?

I am looking for various arguments like

- performance
- administration effort
- cost
- etc

I understand that the maximum number of channels for MQ MVS is 800. Does a client-server channel take up one or two of these?

Answer

First point is that the “limit” of 800 is not fixed - it depends on the MVS configuration. Each active channel consumes 4K of CSA below the line. Each concurrently active client is one channel. (Or more precisely, each active MQCONN is one channel. It is possible that one workstation could have more than one MQCONN active at one time, depending on application design and user behavior.) You need to check how much CSA below the line is available. The limit could be a lot less than 800.

Each channel/client requires 8K of storage in the private area below the 16MB line in the Chinit address space due to a C limitation.
The size of this private area depends on how much MVS uses and how big a CSA is used. You do not have much control over the amount MVS uses, but you do over the size of the CSA. So with a larger CSA then there is a smaller user region which will reduce the number of attachments to the CHINIT address space.

The figure of 800 clients is quoted to set the expectation of people who have a large CSA.

The administrative effort associated with running clients directly into MQM/ESA must be less than maintaining a set of MQM servers. The cost question is the cost of Client Attach Feature on MQM/ESA versus cost of a number of MQM "extended" distributed licences, plus possibly the cost of the servers themselves, unless they already exist.

The major advantage of running clients off local distributed servers is that the clients do not need MVS and the network to MVS to be available in order to make MQ calls. One would hope a local LAN-attached server would have higher availability than MVS across a WAN. If the clients would be LAN-attached to MVS in any case, this argument is weaker.

Performance, of MQCONN in particular, would probably be better with a local server versus going to MVS across a WAN.

MQ Subsystem objects on a sysplex

Question

I'm creating multiple MQ subsystems in a sysplex and was wondering how the queues(for example SYSTEM.COMMAND.INPUT) are differentiated between subsystems, since these are defined by CSQINP2 when CSQxMSTR is started and these admin and command queues have to be specific names?

Answer

When a queue (be it a SYSTEM.* queue or one of your own) is defined, it is for a specific queue manager. If you have more than one queue manager, you have to define these queues for each queue manager. The definition of that queue resides on pageset zero of that queue manager, and the storage class used in the queue definition will determine which pageset the messages put to that queue will reside on (for persistent messages). You cannot share pagesets between queue managers, so the queues defined on each queue manager cannot be used by other queue managers unless they put messages to them via distributed queueing.

There is what you might call a 'fully qualified name' for any Queue defined to MQ. When you setup RACF, this becomes important. so for Qmgr CSQ1, queue SYSTEM.COMMAND.INPUT will be protected by profile CSQ1.SYSTEM.COMMAND.INPUT in class MQQUEUE, and there will be a similar profile for SYSTEM.COMMAND.INPUT in CSQ2.

Whenever you enter a command on a queue manager, you direct that command to a specific queue manager. On MVS, this direction can be achieved a number of ways, for example:

- if you enter the command via CSQINP2, it is directed to the queue manager which processes that CSQINP2 DD
- if you enter the command via the TSO panels (CSQOREXX), you enter the queue manager on the front panel
- if you enter the command via the console, you prefix the command with the CPF of the queue manager you want the command to go to
- if you enter the command via an application, your app will connect to the queue manager specified in the MQCONN call.

When a command is directed to a queue manager, it is processed by the command server on that queue manager. For MVS, this is started when the queue manager is started and remains started unless you issue the STOP CMDSERV command.

You can set up queue and channel definitions to route commands to other queue managers; however, you must still have command queues on your local queue manager or else you wouldn't be able to define the queues and channels to route other commands elsewhere!

In summary, therefore, the command queues are differentiated between queue managers by the combination of each queue manager having its own set of these queues on its own pagesets, and the queue manager the command is directed to.

This is true for both sysplex and non-sysplex setups.

**MVS Performance Considerations**

**Question**

Are there some guidelines and/or considerations for performance and tuning in an MVS environment when a single queue is used?

**Answer**

1. Ensure that you have a sufficient number of buffers allocated to your buffer pools. The default is 1050 buffers. Try increasing this to 50000 or so. Also check your log buffer sizes.

2. How many server applications do you have reading messages from the single queue on MVS? One or more? When you notice that response times are degrading, check the depth of this queue. If it is high, then this will indicate that the requests are arriving faster than your server can handle them, in which case, you should consider increasing the number of servers.

3. What are the sizes of your request and reply messages? Have a look at the MQSeries for MVS/ESA Capacity and Planning report (available on the MQSeries website as SupportPac MP11). This report considers the affects of changing the message sizes.

4. What batch size have you specified on your channel definitions? Again, the above report goes into the affects of varying the batch size.

5. What type of triggering do you use? TRIGTYPE FIRST or EVERY?

6. Are you running with internal trace ON? Try turning trace OFF.
7. Are your messages persistent or non-persistent? What type of requests are you submitting? Queries or Updates? If you are simply submitting queries, then you might consider using non-persistent messages. This will reduce logging overhead.

8. Are the gets and puts done in syncpoint? If you do not care about atomicity, then it would probably be safe to issue your gets and puts outside of syncpoint. But if you do want the ability to back-out, i.e. During abnormal termination, then you should issue your gets and puts in syncpoint.

Queue Manager Name

Question

We have separate development, system test, and production environments. Each of these MVS machines will have its own uniquely named queue manager. However, there is no guarantee that only one queue manager will be running on each system, nor that the queue manager that we use will be the default. We want to be able to make the queue manager name a variable. The applications are realtime. Is our only option to create a "file" that for each system contains a record with the queue manager name to use?

Answer

I would generate three CSQBDEFV modules, each naming one of the three queue managers and put each module in a load library that would be used by applications that connect to the respective queue manager.

MQCONN loads CSQBDEFV at execution time to determine the "default" queue manager. If you create three of these modules and have test, system test and production arrange different version of the module, then each application will automatically pick up the correct version by specifying the default "" queue manager.

You would use RACF to protect the production and system test queue managers against "accidental" connection.

Using MQ to initiate a batch job

Question

MQ can trigger a CICS transaction off on MVS, but can it trigger off a batch job (submitting job or somehow spool a job)?

Answer

I think you need to look into the batch monitor example supplied in MA12.

Unix

AMQ8145 Connection broken / HP-UX

Question

We have just installed MQ on a HP Unix. I have not worked on ANY Unix machine before, and my counterpart on the Unix doesn't know anything about MQ, so.... After installing MQ from the CD we did the following steps:

1. crtmqm -q ODIN
2. strmqm ODIN
3. runmqsc < amqscoma.tst > list

when we looked in the LIST file it said AMQ8145 Connection broken. I then tried the following

1. strmqm ODIN - response was Qmgr already started or running
2. runmqsc <enter> - I now had the console
3. dis q(*)

again after some seconds I got the AMQ8145 Connection broken. Anyone seen this before?

Answer

Have you updated your kernel configuration to match the values in the System Administration Guide?

If that doesn't solve your problems, are there any .FDC files in /var/mqm/errors?

Answer

No we have not, but will do so now.

MQSeries V5 now supports HP-UX V11

News

MQSeries are pleased to announce MQ/V5 support for HP-UX opsys V11. The following text is restatement of Document APAR IX77529 which accompanies this support statement, giving use guidance and explaining full details of the operational terms of the support.

MQSeries for HP-UX, V5.0 can run on HP-UX V11

However, due to the operating system differences between HP-UX v10 and v11, this APAR has been raised to document the way in which this can be done.
Any application compiled and linked on HP-UX v10.10 or v10.20 can run on HP-UX v11, provided that ALL its associated applications are also compiled and linked at v10. (i.e. fully bound).

If v10 applications try to call or be called by v11 applications, the results are unpredictable and this configuration is not supported by Hewlett-Packard.

As MQSeries is an application that was compiled and linked on HP-UX v10, it is possible to execute these binaries on v11, provided all the user applications are also compiled and linked on v10 machines and copied over to the v11 machine.

The reason for this is as follows.... On HP-UX v11, two versions of libc.sl have been supplied with the opsys. The version that the application uses depends on what machine the application was compiled and linked on. These two versions are libc.1 and libc.2. When an application is compiled and linked with libc.sl on v10 machine, the application will actually use libc.1 when run on the v11 machine. However, if the compilation, linking and execution occurred on the v11 machine, the application will use libc.2. Therefore, if users compile and link applications that make use of the MQSeries V5.0 API on v11 machines, the applications will fail with unpredictable errors.

Installation changes for HP-UX V11.

Within the SAM fileset install window, the option "allow incompatible software" must be checked so that the files will actually be written to the hard disk. Install then proceeds as normal, but terminates with the following warning.

WARNING: The product "MQSERIES,r=B.10.500.00" will not be configured because it is not compatible with this system. Configuration can only be done on compatible systems, or by running the "swconfig" command with the "allow_incompatible" option set to "true".

The following then needs to be run from the command line to fully configure MQSeries to run properly:

```
swconfig -x allow_incompatible=true MQSERIES
```

Also, the MQSeries Quick Beginnings Guide (GC33-1869-00) shows information not present in the HP-UX 11 kernel;

Chapter 4. Installing the MQSeries for HP-UX Server Kernal Configuration Figure 1.

"The following two alterable parameters as listed:

```
semmsl   100
semopm   100
```

**Error AMQ9523 and AMQ9207 in LOG**

**Question**

We have installed MQSeries on two AIX 4.2 machines. Before we doing anything else, we wanted to make sure everything worked properly. We set up a unidirectional queue where one machine is the sender and the other is receiver. This worked fine going in one direction, but when we reversed the roles of the two machines we got an AMQ9523 "bad protocol" error when pinging from the sndr. In the LOG file, the error was AMQ9523.
What could we be doing wrong? Everything seems to be installed properly, and is installed the same on both machines.

**Answer**

Error message AMQ9207 suggests you have something wrong with your configuration on the receiving machine. There should be an FFST generated and in it there will be some text which is the first few bytes we received back from the (failing) AIX machine. Go have a look on the machine you tried the ping from.

**MQ on AIX**

**Question**

We have installed MQSeries Version 5.0 on AIX 4.2. When attempting to verify Server to Server installation (p32 of the "Quick Beginnings" manual) we get an AMQ9207 "The data received from host <host name and IP address> is not valid". This message is issued after attempting to start the sender channel by executing the command:

```
runmqchl -c first.channel -m saturn.queue.manager
```

Can anyone help?

**Answer**

My guess is that what's happening here is that you have not configured inetd correctly on your receiving machine. When inetd receives a connection and starts the program (amqcrsta), it does so with the socket dup'd over stdin and stdout. This means that during the first phase of the connection anything written to stdout (ie. an error message) will go down the socket to the runmqchl program. It's worth checking the runmqchl side to see whether an FFST is generated with the invalid data in it. If you're lucky you'll get the print out of the error message generated by inetd or amqcrsta. You might just get an error number.

Either way, I'd check you inetd.conf file (are you sure you entered two amqcrsta words in the line?)

**N0 Default QManager (AIX)**

**Question**

We would like to work with a No Default QManager in AIX 4.1 (MQS 2.2.1. with LU6.2 (AIX SNA Server 3.1), connected to a MQS/MVS 1.4 (CICS Mover).

The Distributed Queuing Guide (page 259: Receiving LU6.2) states that we have to add in the TPN Profile 'Use Command Line: Yes' and 'Command Line Parameters: -m QManager_Name'.

If the QManager is defined as Default (mqm.ini), all works o.k., but if the QManager is NO Default (there is another Default QManager), when we send a Ping from MVS (o start the Sender Channel), we obtain the
CSQK546I msg (Remote QManager Unavailable). Also in the @SYSTEM error log (AIX), the msg AMQ9508 Cannot Connect to the queue Manager appears.

**Answer**

When you want to connect to Non-default Qmgr using SNA, you have to make a simple script like:

```bash
#!/bin/ksh
amqcrs6a -m qmgr
```

and in TPN profile of SNA, put the name of this script instead of amqcrs6a. Then you will be able to connect to Non-default Qmgr.

---

**Permission problem**

**Question**

I have just installed MQSeries on AIX. The script to test the implementation works fine as long as you are logged in as root or as mqm. I have created the mqm group and I am a member of the group, but whenever I try and control commands or mqsc commands the only message that I get is the following: MQSeries was unable to display an error message.

What is wrong with the installation?

**Answer**

You may try looking at the permissions in the /usr/lpp/mqm and /var/mqm directories as well as the permission on runmqsc (it should be SETUID,SETGID to mqm). If for some reason these have been changed that would produce symptoms like you are describing.

---

**AIX MQseries QA**

**Question**

Does AIX MQseries support multiple servers reading or writing one queue in parallel? We want to write an application where multiple clients write to one message queue and multiple servers process those messages.
Answer

Yes, on AIX you can have multiple servers for a queue. It's only on DEC that you're restricted to a single server for a queue.

All platforms support multiple applications putting to a single queue.

AIX SIGALRM restriction

Question

The AIX System Management Guide recommends that applications should not use SIGALRM in applications that connect to MQSeries queue managers.

This seems to be a problem in applications that need to implement a timer -- and that would like to use SIGALRM for that purpose. I don't yet fully understand why that particular customer thinks he needs to have a timer in his MQSeries applications and I will discuss this further with the customer.

However, can please somebody explain how exactly the "it is not recommended" in the System Management Guide should be interpreted -- that is, what could go wrong if we tried to ignore that recommendation?

Perhaps it would help if we would understand in more detail how and for what purpose MQSeries uses the SIGALRM signal handler.

If the customer insists that he needs to have his own timer -- is there a way to implement a timer without disturbing (and without being disturbed by) the MQSeries use of SIGALRM?

Answer

The situation in AIX is that there is one timer per process. This can be set such that it will raise SIG_ALRM after a specified time interval has expired. If the application has programmed a timer interval and has set a SIG_ALRM handler to catch it, it should avoid making ANY calls outside its own control (eg. MQ calls). If a timer is pending when an MQ call is issued, then MQ will reprogram the timer for its own use, and all knowledge of the previous timer will be lost.

There is no problem with the application using the timer, outside of issuing MQ calls, since MQ will reset the environment each time as required. However, the restriction is that there should be no pending SIG_ALRM timer requests when issuing an MQ call.

Kernel Configuration on Sun Solaris 2.2

Question

In the MQSeries for Sun Solaris System Management Guide on Page 17 Figure 1 are the minimum initial recommended values of the kernel parameters listed. The customers has a given installation where the parameters
SEMMNS = 640 (8000 recommended for MQSeries) and
SEMOFM = 10 (100 recommended for MQSeries).

His question is will MQSeries operate with his chosen parameters, or must he change his parameters to the recommended ones. What will happen if he operates with his given parameters?

**Answer**

He must change his parameters to the recommended ones. If he does not, sooner or later the queue manager will start spitting out FFST's (internal errors). This may be immediate or it may be after the queue manager has been running for a while. Sorry, he needs to change those parameters.

**MQSeries for AIX and CCSID**

**Question**

Customer wants to change the CCSID used by MQSeries for AIX. A change of the primary language in the AIX environment didn't show any effect on the CCSID used by MQSeries. Also a change of environment variable LANG to En_US didn't make MQM to use the desired codepage 850.

After reading the manuals the following questions arose: Where (from what AIX environment information) does MQM read the CCSID when creating a Queuemanager by running CRTMQM? How can we set the CCSID when creating the Queuemanager? Is there a possibility to change the CCSID when the Queuemanager is already created?

**Answer**

The CCSID is picked up from the operating system when you do a crtmqm. It is NOT possible to alter CCSID after the Qmgr has been created.

'export LANG=En_US' should set the CCSID of the next queue manger to 850.
'export LANG=en_US' should set the CCSID of the next queue manger to 819 (= ISO8859-1).

**MQSeries for AIX and DCE threads**

**Question**

What is the scope of a connection handle in this environment?

1. the process
2. the dce thread

Or in other terms,

- Is it allowed to pass a connection handle from one thread to another? (in the middle of a transaction).
What transaction gets committed when an MQCMIT is executed? All MQI instructions in the thread or process.

Is the scope of a connection handle the same as the scope of an MQI transaction.

Can different threads of the same process connect to different queue managers at the same time?

When a thread ends, does a connection created in that thread get closed (cleaned up) automatically, just as when the process ends?

**Answer**

The scope of a connection handle is the process (in non-DCE) or the DCE-thread (in DCE). As a rule of thumb in any operating system, the scope of the handle is a thread of execution. So you cannot connect in one thread and obtain a handle and then pass the handle to any other thread or process. It is only usable by the original calling thread.

So for the rest of your questions:

- No, you can't pass a connection handle from one thread to another
- The transaction that gets committed by MQCMIT is the one associated with the connection handle, i.e. all syncpointed work that was performed using that connection handle. This implicitly restricts the scope to the calling thread.
- I don't think different threads can talk to different Queue Managers.
- At present when a DCE thread ends the connection is not cleaned up since we cannot detect it. Cleanup will occur when the process ends.

**Questions on MQ recovery**

**Question**

Some basic questions on MQSeries recovery. In this environment, All MQ data (indeed, all data on the system) is mirrored on RAID 1 DASD.

We're building a system which we can shut down every night, perform backups, etc. I'd like to keep the system as operationally simple as possible. Hence, the following simple questions:

1. If I shut down MQ, backup the queues and logs (and any other data necessary), then later start MQ again, does MQ automatically recover queue data if the queue is damaged if I'm using CIRCULAR logging?

2. Can MQ recover if the log is damaged, but the queues are not? Is this true for both circular and linear logging? If so, what do we have to do?

3. If I'm using linear logging, can I recover the queues to a time PRIOR to the most recent checkpoint (syncpoint)? Could I, if I had created the proper media recovery records for all the queues in question? In certain of our recovery scenarios, we may have to restore the database similarly, and I would like the queues to be synchronized with the database contents at that time.

4. Can I use dual logging in an AIX environment? Some of the documentation says MVS, other implies AIX also.
5. How can I find out which (linear) logs can be archived/removed programmatically, without having to watch for operator messages constantly?

Answer

1. No. Damaged queues are never recovered if circular logging is being used. If you back up the whole system, damage a queue, and then recover the whole system (outside of MQ) then MQ should never know about it so the MQ logging should not be relevant.

2. No. Go to your last backup. This is true for both circular and linear logging.

3. No. Recovery will be to the time at the end of the log. So, you can recover to the time of a particular backup. You can't bound the recovery processing to a particular point before the end of the log.


5. I'm afraid you can't. You would have to write a program to watch the operator message logs.

TPname for AIX to MVS using LU 6.2

Question

We are trying to setup channel communications between MVS and AIX using MQSeries and native DQM without CICS.

It is documented to specify a TPname in the side info in AIX but that the ASCH scheduler and TP profile dataset are not used on MVS. Can anyone explain how the TPname is used on MVS then if no profile is defined.

We specified APINGD on AIX side info and MVS side info. The channel starts ok from MVS to AIX but from AIX we get CPI-C RC 10. The listener is active on MVS and APPC session up between the workstation and MVS.

Answer

MQSeries for MVS/ESA doesn't use ASCH; it uses an APPC/MVS server and therefore doesn't require TPDATA.

The Listener registers to APPC/MVS to receive inbound allocates for the LU/TP combination specified in the listener side object. When a remote system allocates to APPC/MVS, the allocate is passed to the listener who starts a responder channel.

Map the LUNAME parameter value specified on the START LSTR command to the side info dataset specified in SYS1.PARMLIB(APPCPMxx). The Partner Lu and TP Name for this side object will be those that you should specify to connect to the listener from a remote system.

Trigger on HP-UX
**Question**

I need some assistance: I am unable to start the MQ-Series trigger monitor under a different ID than mqm on our Unix platform. This again has other implications with regards to running other processes. 200 MQI clients - running on NT boxes (connections via dial-up line).

**Answer**

The standard trigger monitor is shipped setuid/setgid to mqm. You also have to be in the mqm group to start it running. Any processes it starts (unless they in turn are setuid) will run with mqm authority.

There is nothing stopping you copying the runmqtrm program to another name, changing its permissions, and running it as some other ID provided you understand the security implications of doing so. That other ID would of course have to have authority to read the initQ. But one potential security hole is then that an mqm administrator could set up a process definition to run any other program under your new trigger monitor with that userid's authority. Perhaps that is a problem for you - I wouldn't normally allow it on my machines.

Only root-authorised programs can change to an arbitrary userid on Unix systems. But you probably don't want a trigger monitor running arbitrary programs (configured by mqm-admins) under any userid including root.

It comes down to how much you trust what MQ and other system administrators do on that box. And none of this is really an MQ issue - it's much more to do with regular Unix security controls.

Because you can write your own trigger monitor programs (we even ship source code as a sample to start from) you could write one which only allowed certain operations to certain userids and then that trigger monitor could be setuid root. For example, it could ignore the applid or use it to look up in another config table the real program/authority to run with. You'd have to be very careful about it, but it is possible. There is no generic solution however, because these things are site-specific.
Sending your comments to IBM

MQSeries Knowledge Base
Version 1.0

MA0B SCRIPT

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book. Please limit your comments to the information in this book and the way in which the information is presented.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate, without incurring any obligation to you.

You can send your comments to IBM in any of the following ways:

• By mail, use the Readers’ Comment Form.
• By fax:
  – From outside the U.K., after your international access code use 44 1962 841409
  – From within the U.K., use 01962 841409
• Electronically, use the appropriate network ID:
  – IBMLink: IBMGB(TSCC)
  – Internet: tsc@hursley.ibm.com

Whichever you use, ensure that you include:

• The publication number and title
• The page number or topic to which your comment applies
• Your name and address/telephone number/fax number/network ID.
MA0B SCRIPT

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. This form is provided for comments about the information in this manual and the way it is presented.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Be sure to print your name and address below if you would like a reply.

Name

Company or Organization

Telephone

Address

Email

--

--
MQSeries Knowledge Base
MA0B SCRIPT

You can send your comments POST FREE on this form from any one of these countries:

Australia   Finland   Iceland   Netherlands   Singapore   United States
Belgium     France     Israel     New Zealand     Spain       United States of America
Bermuda     Germany    Italy      Norway        Sweden      Switzerland
Cyprus      Greece     Luxembourg Portugal       Switzerland
Denmark     Hong Kong  Monaco    Republic of Ireland United Arab Emirates

If your country is not listed here, your local IBM representative will be pleased to forward your comments to us. Or you can pay the postage and send the form direct to IBM (this includes mailing in the U.K.).

By air mail
Par avion

IBRS/CCRI NUMBER: PHQ - D/1348/SO

IBM

REponce PAyee
GRANDE-BRETAGNE

IBM United Kingdom Laboratories Limited
Information Development Department (MP 095)
Hursley Park
WINCHESTER, Hants
SO21 2ZZ

United Kingdom

From: Name ___________________________________________
Company or Organization ______________________________
Address ____________________________________________
EMAIL _____________________________________________
Telephone __________________________________________