

Instructions and Warnings: How to increase the MAXMSGL for IBM MQ Queue Managers, Queues and Channels from the default 4 MB to a higher number

<https://www.ibm.com/support/pages/node/400151>

Date last updated: 20-Apr-2023

Angel Rivera
IBM MQ Support

<https://www.ibm.com/products/mq/support>
Find all the support you need for IBM MQ

+++ Problem

You have an IBM MQ Client application that works fine in sending short messages to an MQ queue manager. However, whenever the size of the message is greater than 4 MB, you get errors with reason codes such as:

2010 MQRC_DATA_LENGTH_ERROR
2031 MQRC_MSG_TOO_BIG_FOR_Q_MGR
2218 MQRC_MSG_TOO_BIG_FOR_CHANNEL

++ There are 4 main sections in this article:

- a) Notes and recommendations
- b) Caveats and Warnings
- c) Few objects to support maximum message length of 100 MB.
- d) All objects to support a maximum message length of 100 MB.

++ Update on 26-Jan-2022

Thanks to Morag Hughson and Bob Gibson for their suggestions to improve this document.

+++ Section A: Notes and recommendations

++ Overall limit

The MAXMSGL specified at the Queue Manager Level is the TOP Limit for ALL objects.

<https://www.ibm.com/docs/en/ibm-mq/9.3?topic=reference-display-qmgr-display-queue-manager-settings>

IBM MQ / 9.3

DISPLAY QMGR (display queue manager settings)

+ begin excerpt

MAXMSGL

The maximum message length that can be handled by the queue manager. Individual queues or channels might have a smaller maximum than the value of this parameter.

+ end excerpt

++ Notes:

+ The attribute MAXMSGL is the maximum size in bytes for a message, which includes the Message Descriptor and the Payload.

+ The size of the message descriptor (MD) is variable depending on certain things such as persistent or not persistent, special headers such as RHF2 for JMS, etc.
For planning purposes, a "rough" estimate would be around 2,000 bytes to account for a large MD.

+ The default limit for maximum message size for MQ channels and MQ queues is:
4 MB or 4194304 or 4 194 304 bytes

+ The upper limit is:
100 MB or 104857600 or 104 857 600 bytes

+ If you are using the MQSERVER environment variable which takes precedence over the Channel Client Definition Table (CCDT), the limit in MQ V6 was 4 MB but it was raised to 100 MB in MQ V7.

+ The MAXMSGL specified at the Queue Manager Level is the TOP Limit for ALL objects. Thus, if any object has 8 MB, for example, then the queue manager ALSO needs to be updated to 8 MB.

+ The utility dmpmqmsg can handle messages with the maximum message size of 100 MB.

+ If a server-connection channel is defined with MAXMSGL 10 MB but the client-connection channel is defined with a smaller value such as 9 MB, then the negotiated value is the LOWEST, that is 9 MB. And vice versa, server-connection channel with 9 MB but client-connection with 10 MB, the negotiated value is the LOWEST, that is 9 MB.

+ The active instances of objects, such as open Queues and active Channels, will NOT be affected by the change, because when they were opened/started they used the old value. Thus, after making the change, you will need to close/reopen queues and to stop/start channels and to disconnect/reconnect the client application in order for the newer instances to use the new values.

++ Related return codes

<https://www.ibm.com/docs/en/ibm-mq/9.2?topic=codes-2010-07da-rc2010-mqrc-data-length-error>

IBM MQ 9.2.x / IBM MQ / Troubleshooting and support / Reason codes and exceptions / API completion and reason codes / API reason codes /
2010 (07DA) (RC2010): MQRC_DATA_LENGTH_ERROR

<https://www.ibm.com/docs/en/ibm-mq/9.2?topic=arc-2031-07ef-rc2031-mqrc-msg-too-big-q-mgr>

IBM MQ 9.2.x / IBM MQ / Troubleshooting and support / Reason codes and exceptions / API completion and reason codes / API reason codes /
2031 (07EF 0x000007ef) (RC2031): MQRC_MSG_TOO_BIG_FOR_Q_MGR

<https://www.ibm.com/docs/en/ibm-mq/9.2?topic=arc-2218-08aa-rc2218-mqrc-msg-too-big-channel>

IBM MQ 9.2.x / IBM MQ / Troubleshooting and support / Reason codes and exceptions / API completion and reason codes / API reason codes /
2218 (08AA - 0x000008aa) (RC2218): MQRC_MSG_TOO_BIG_FOR_CHANNEL

++ Recommendations

- If you need to increase the MAXMSGLEN for a Queue, you will need to increase MAXMSGLEN for all the objects that are in the path for the flow of the message, for example, the server-connection channel.
- If you only increase MAXMSGLEN for the Queue, but not for the corresponding channel, then you will not get the error that the message is too big for the queue, but you will get the error that the message is too big for the channel.
- Set MAXMSGLEN to a value that is a bit larger than the largest size that you expect to work with. If the largest size is 7 MB, then update MAXMSGLEN to 8 MB.
That is, due to the caveats and warnings, we do NOT recommend that you increase the size to the maximum of 100 MB (it might be an overkill).
- If you plan to increase MAXMSGLEN to a big value, then it is VERY important to be aware of the several CAVEATS described in this technote.

++ Is it possible to handle large objects using the MQ infrastructure? Answer: YES, with MFT

+ Can "MQ Segmentation" be used to overcome the 100 MB limit? Answer: NO

A common (but incorrect) attempt to circumvent the limit of 100MB is to try to use the feature "MQ Segmentation": it will NOT work.

For example, you prepared 3 segmented messages of 40 MB (for a total of 120 MB), so far, so good. But when the application that is handling the messages from the queue tries to reassemble the 3 segmented messages will do this:

- A buffer will be created to reassemble the segments. So far, there are 0 MB in the buffer.
- The 1st segment (of 40 MB) is read and added successfully to the buffer. Now there are 40 MB in the buffer.
- The 2nd segment (of 40 MB) is read and added successfully to the buffer. Now there are 80 MB in the buffer.
- The 3rd segment (of 40 MB) is read, but when trying to add it to the buffer (which has already 80 MB), it will fail because the total size would be 120 MB which is beyond the maximum limit of 100 MB

++ For large objects, you could consider treating them as files and use MQ MFT

To handle messages that are larger than 100 MB, convert them into files, and then use IBM MQ Manage File Transfer (MFT) to transfer the files.

MQ MFT breaks down a large file into packets that are smaller than the MAXMSGL, then transmits those packets to another queue manager, where they are reassembled by an MQ MFT agent.

Note: MFT does not use the feature of "MQ Segmentation". MFT uses another method.

There is no hardcoded upper limit in MQ regarding the size of the file. Files with several GBs can be sent. The limits are the ones imposed by the system resources.

The use of MQ MFT requires an additional license.

The MQ MFT installation code is included with the MQ package that is obtained from IBM Passport Advantage.

For more information see:

<https://www.ibm.com/docs/en/ibm-mq/9.3?topic=overview-managed-file-transfer>

IBM MQ / 9.3

Managed File Transfer

+++ Section B: Caveats and Warnings!

Keep in mind that increasing the maximum message length from 4 MB to a high value such as 100 MB will impact many tuning parameters of the queue manager. Improper tuning may result in exhaustion of server resources which in turn will negatively impact the queue manager, including possible failure. Tuning parameters affected by MAXMSGL include the following:

- * Channel batch sizes are expressed as a number of messages, regardless of the message size. With default settings, the channel will not attempt to send more than 200MB in a single batch. Since batches are transactional, this also means that no more than 200 MB is held under syncpoint for any given channel with default settings. With MAXMSGL set to 100 MB and a channel batch size of 50, the largest possible channel batch size is 5 GB. This can cause wide variation in channel performance when large and small messages are mixed on the same channel, or even total failure of the channel (see the bullet on log file tuning below). If possible, segregate very large and very small messages onto separate channels that are tuned appropriately.

- Note (new in 26-Jan-2022, Thanks to Morah Hughson)

One way to mitigate the situation is by setting the attribute BATCHLIM: This attribute is the limit, in kilobytes, of the amount of data that can be sent through a channel before taking a sync point. A sync point is taken after the message that caused the limit to be reached has flowed across the channel.

If you just keep with the default value of BATCHLIM which is 5000 kilobytes, so around 5MB, you will never have 5GB of data in one transaction. With the default BATCHSZ and BATCHLIM, if you send a single 100MB message across the channel, it would immediately close the batch having exceeded the BATCHLIM number, so you would not be able to get 50 messages of that size in a single batch.

- * MAXUMSGS is a queue manager parameter that limits the maximum number of uncommitted messages a process may hold under syncpoint before a rollback of the transaction is forced. Each of these messages consumes disk space, memory and some portion of log file active extents. Raising MAXMSGL to 100MB allows the same number of messages under syncpoint to consume up to 25 times more of these resources. Depending on the volume of very large messages, it may be advisable to tune MAXUMSGS to a lower value.

- * All data held under syncpoint at any given moment must fit within the primary and secondary log file extents. This is true even when linear logging is used. The default log file settings provide for 80 MB of data total, for all processes including internal queue manager processes such as channels, under syncpoint at any moment. Log file size and extents must be tuned according to the number of very large messages that are expected to be in flight at any moment. Changing the number of log file extents requires a restart of the queue manager. Changing the size of the log file extents requires deleting and rebuilding the queue manager so it is advisable to perform this tuning before the queue manager is created.

* The default MAXMSGL is designed to ensure that the maximum disk space for a queue fits within the 2 GB limit of older file systems. In some cases, it may be necessary to tune MAXDEPTH to a lower value to maintain this limit. If the queue exceeds the maximum file size for the underlying file system applications attempting to PUT a message, receive a return code indicating a resource error rather than the expected QFULL or MSG_TOO_LARGE.

* When MAXMSGL is set to 100 MB it is much more likely that the underlying file system will fill before the application hits MAXDEPTH on the queue. When an application attempts to exceed MAXDEPTH that application receives a QFULL error code and can retry the PUT. However, if the file system fills the entire queue manager and all connected applications are prevented from putting any new messages. It is always advisable to ensure adequate disk storage under the queue and log files and this is especially true when tuning for very large messages.

* Queue manager restart times vary based on the amount of data queued up and under syncpoint. This is true for stand-alone as well as multi-instance and hardware clustered queue managers. Thus, there is a trade-off between the amount of data that can be queued versus the speed of recovery after failure. If restart time is critical and very large messages are common, it may be necessary to tune MAXDEPTH and log file capacity to lower values.

* The strategy of the default values for MQ is to return soft, recoverable errors to applications rather than to allow exhaustion of underlying server resources such as disk space or memory. When raising any of these soft constraints, it is advised to consider the possible effects and whether to compensate by lowering the value of a corresponding tuning parameter.

* MAXFSIZE (new in 26-Jan-2022, Thanks to Morah Hughson)

Starting with MQ 9.2 you can control the size of queue files using an attribute on local and model queues.

MAXFSIZE => Denotes the maximum size of the queue file used by the queue, in megabytes. This queue attribute MAXFSIZE does a similar sort of job as BATCHLIM to batches, for the queue file on queues.

* The message length has an impact in the recover logs used by MQ and is related to the maximum depth of the queue and persistent messages. Please see the following technote:

<https://www.ibm.com/support/pages/node/241741>

How to calculate the total amount of disk space for the recovery logs for one queue manager

+++ Section C: Few objects to support maximum message length of 100 MB

Scenario: An application that uses the MQ Client is connecting via a server-connection channel and accessing a queue.

You need to perform the following configuration changes in MQ to ensure that 100 MB messages can get through from the MQ Client through the server-connection channels to the Queue.

Notes regarding Clustering and Distributed Queueing:

- If your scenario uses MQ Clustering, then you need to ensure that the corresponding queues (SYSTEM.CLUSTER.TRANSMIT.QUEUE) and channels CLUSSDR/CLUSRCVR involved with the Cluster are modified too.
- If you are using a Remote Queue Definition in a local queue manager, then you will need to ensure that the Transmission queue (XMITQ), the channel SENDER/RECEIVER and the destination queue in the remote queue manager are modified too.

Start by invoking the MQ administration tool:

```
runmqsc QMgrName
```

```
# Display the maximum message length for the Queue Manager.
```

```
# The default is 4 MB.
```

```
# In this example, the name of the queue manager is QM_VER.
```

```
display qmgr maxmsgl
```

```
1 : display qmgr maxmsgl
```

```
AMQ8408: Display Queue Manager details.
```

```
QMNAME(QM_VER) MAXMSGL(4194304)
```

```
# Alter the value to the upper limit 100 MB
```

```
alter qmgr maxmsgl(104857600)
```

```
# Display the maximum message length for the queue.
```

```
# Using the name Q1 as example where the MQPUT is used to put a message.
```

```
# The default is 4 MB
```

```
display ql(Q1) maxmsgl
```

```
4 : display ql(Q1) maxmsgl
```

```
AMQ8409: Display Queue details.
```

```
QUEUE(Q1) TYPE(QLOCAL)
```

```
MAXMSGL(4194304)
```

```
# Alter the value to the upper limit 100 MB
```

```
alter ql(Q1) maxmsgl(104857600)
```


Display the maximum message length for the server connection channel.

Using as example: SYSTEM.AUTO.SVRCONN.

The default is 4 MB

display CHANNEL(SYSTEM.AUTO.SVRCONN) MAXMSGL

9 : display CHANNEL(SYSTEM.AUTO.SVRCONN) MAXMSGL

AMQ8414: Display Channel details.

CHANNEL(SYSTEM.AUTO.SVRCONN) CHLTYPE(SVRCONN)

MAXMSGL(4194304)

Alter the value to the upper limit 100 MB

Notice that the proper CHLTYPE must be specified after the channel name.

alter CHANNEL(SYSTEM.AUTO.SVRCONN) CHLTYPE(SVRCONN) maxmsgl(104857600)

Display the maximum message length for a Client Connection channel

(when using CCDT).

In this example, it is the custom channel named "CH1":

display channel(CH1) chltype(clntconn) maxmsgl

16 : display channel(CH1) chltype(clntconn) maxmsgl

AMQ8414: Display Channel details.

CHANNEL(CH1) CHLTYPE(CLNTCONN)

MAXMSGL(4194304)

.

Alter the value to the upper limit 100 MB

Notice that the proper CHLTYPE must be specified after the channel name.

alter channel(CH1) chltype(clntconn) MAXMSGL(104857600)

17 : alter channel(CH1) chltype(clntconn) MAXMSGL(104857600)

AMQ8016: MQ channel changed.

End the session

end

+++ Section D: All objects to support a maximum message length of 100 MB

This section is also when a message is sent from a Queue Manager to another Queue Manager.

In this case, the changes **MUST** be done in BOTH queue managers.

If you want all of your channels and queues to support 100 MB message length then you need to alter all existing definitions, and also alter the system defined objects, in that way, new definitions of channels and queues are defined with the maximum message length set to 100 MB.

In addition to the objects mentioned in (a), you will need to change more objects, such as:

Channels:

SYSTEM.DEF.* (such as SYSTEM.DEF.SVRCONN, SYSTEM.DEF.SERVER)

Queues:

SYSTEM.DEFAULT.LOCAL.QUEUE

SYSTEM.DEAD.LETTER.QUEUE

SYSTEM.DEFAULT.ALIAS.QUEUE

SYSTEM.DEFAULT.INITIATION.QUEUE

SYSTEM.DEFAULT.MODEL.QUEUE

SYSTEM.DEFAULT.REMOTE.QUEUE

SYSTEM.DURABLE.MODEL.QUEUE

SYSTEM.RETAINED.PUB.QUEUE

Start by invoking the MQ administration tool:

runmqsc QMgrName

Alter the value for the Channels and Queues mentioned above.

End the session

end

+++ end