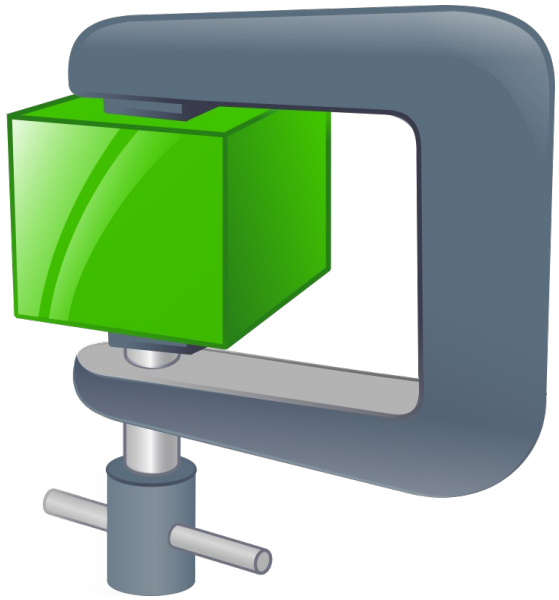
# MQ Message Compression Overview



Capitalware Inc.
Unit 11, 1673 Richmond Street, PMB524
London, Ontario  N6G2N3
Canada
sales@capitalware.com
https://www.capitalware.com

Last Updated: January 2021.

# Table of Contents

# 1 Introduction

## 1.1 Overview

***MQ Message Compression*** (MQMC) provides compression for MQ message data while it resides in a queue or topic and in the MQ logs (i.e. all data at rest). Data compression is the process of modifying and/or converting the structure of bits of data so that it consumes less space in memory and/or on disk.

***Question: Would you trade a little CPU time to drastically reduce the disk I/O time?***

MQMC has implemented 7 lossless compression algorithms:

- ***LZ4*** is a lossless data compression algorithm that is focused on compression and decompression speed. It belongs to the LZ77 family of byte-oriented compression schemes. LZ4 algorithm is incredibly fast. MQMC implemented the LZ4 project.

- ***LZW*** encodes sequences of 8-bit data as fixed-length 12-bit codes. I used Michael Dipperstein's implementation of LZW (Lempel-Ziv-Welch). MQMC implemented LZW from Michael Dipperstein's lzw project.

- LZMA uses a dictionary compression algorithm, whose output is then encoded with a range encoder, using a complex model to make a probability prediction of each bit (Lempel–Ziv–Markov). MQMC implemented LZMA SDK from 7-Zip.

  - ***LZMA_FAST*** uses LZMA SDK with a Level set to 4.
  - ***LZMA_BEST*** uses LZMA SDK with a Level set to 5.

- ***RLE*** (Run Length Encoding) encodes sequences of the same data value occurring over many consecutive data elements that are stored as a single data value and count.

- zlib only supports one algorithm, called DEFLATE, which uses a combination of a variation of LZ77 (Lempel–Ziv 1977) and Huffman coding. MQMC implemented zlib from Rich Geldreich's miniz project.

  - ***ZLIB_FAST*** uses zlib with a Level of *Z_BEST_SPEED*.
  - ***ZLIB_BEST*** uses zlib with a Level of *Z_BEST_COMPRESSION*.

Each queue in a queue manager is assigned two buffers to hold messages (one for persistent messages and one for non-persistent messages). The persistent queue buffer size is specified using the tuning parameter ***DefaultPQBufferSize***. The non-persistent queue buffer size is specified using the tuning parameter ***DefaultQBufferSize***.

- DefaultPQBufferSize has a default value of 128KB for 32-bit Queue Managers and 256KB for 64-bit Queue Managers.
- DefaultQBufferSize has a default value of 64KB for 32-bit Queue Managers and 128KB for 64-bit Queue Managers.

Here's the process of the queue manager handling an application putting a message to a queue:

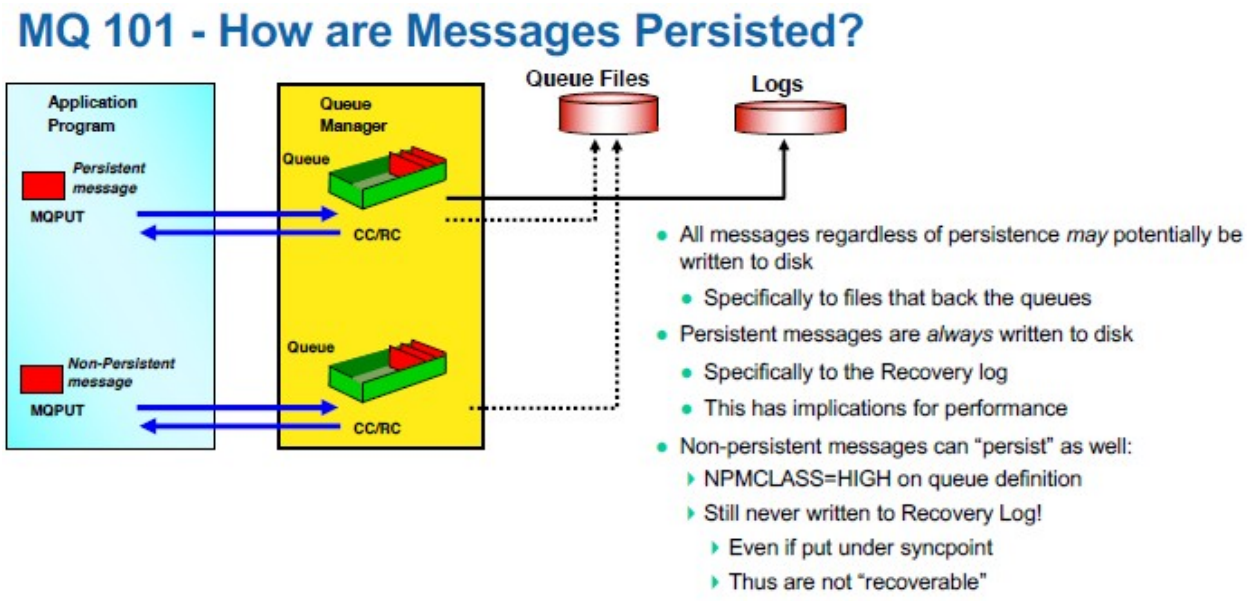
- The message will be put into the buffer of the waiting application if it can fit.
- If that fails, the queue manager tries to write the message to the queue buffer, if it can fit.
- Otherwise, it is written to the queue file.

Here's the process of the queue manager handling an application getting a message from a queue:

- When the consumer (non-waiting) gets a message from a queue, the queue manager will retrieve it from the queue buffer, if available, otherwise from the queue file.
- If the consumer was waiting for a message, then the queue manager will attempt to write it directly to the applications buffer.

## *Persistent messages are always written to the recovery log files.*

Here is a screen-shot from Chris Frank's MQ Technical Conference 2016 session called: More Mysteries of the MQ Logger (page 9) that provides a high-level view of disk I/O.




In the picture, the solid line shows the queue manager writing the messages to the recovery log files. The dotted lines mean that the message may or may not be written to the queue file. See the above for the scenarios of when/why the queue manager would write a message to the queue file.

The goal of MQMC is to improve the disk I/O (Input/Output) speed by compressing the message data when the queue manager is writing to the queue buffers, queue backing files and the recovery log files.

MQMC is an MQ API Exit that operates with IBM MQ v7.1, v7.5, v8.0, v9.0, v9.1 and v9.2 in Windows, Unix, IBM i (OS/400) and Linux platforms.

On AIX, HP-UX, Linux, Solaris and Windows, MQMC can be configured and used with a non-default installation of MQ in a multi-install MQ environment.

Note: Raspberry Pi is a Linux ARM 32-bit OS (Operating System). Hence, simply follow the Linux 32-bit instructions for installing and using the solution on a Raspberry Pi.

MQMC includes an auxiliary program called: testcmprsn. The testcmprsn program allows the end-user to test various types of messages to see which types of messages would benefit from message compression.

Here's an example of testcmprsn program being run against a 9.17MB XML file (huge file):

```
~/test> ./testcmprsn very_lrg_msg.xml
testcmprsn version 1.0.0.0 (Linux64) {Oct 3 2020}

very_lrg_msg.xml size is 9614354 (9.17MB)
Time taken to perform memcpy() is 4.8770ms

Algorithm               Compressed      Compression     Compression     Decompression
                           Size         Time in ms        Ratio          Time in ms
LZ4                  112253 (109.62KB)       3.4830      85.65 to 1         2.9540
LZMA Fast             32872 (32.10KB)      108.4230     292.48 to 1        11.0730
LZMA Best             27675 (27.03KB)     1152.6960     347.40 to 1        10.6730
LZW                  287184 (280.45KB)     203.0840      33.48 to 1        80.8820
RLE                13213500 (12.60MB)       28.1200       0.73 to 1        26.2680
ZLIB Fast            240612 (234.97KB)      28.3140      39.96 to 1        11.2530
ZLIB Best             83375 (81.42KB)       88.5010     115.31 to 1         8.4590
testcmprsn is ending.
```

## 1.2 Executive Summary

MQMC is an MQ API Exit. The MQ API Exit is available in 3 forms:

- Windows DLL
- Shared library for AIX, HP-UX, Linux, and Solaris.
- IBM i exit module

The major features of MQMC are as follows:

- No application changes required
- All message data written to a selected queue and/or topic will be compressed (nothing missed or forgotten)
- Compression/decompression algorithms used: LZ4, LZW, LZMA_FAST, LZMA_BEST, RLE, ZLIB_FAST & ZLIB_BEST
- Standard MQ feature, GET-with-Convert, is supported
- Provides high-level logging capability for compression/decompression processing

## 1.3 Prerequisites

This section provides the minimum supported software levels. These prerequisites apply to server-side installations of MQ Message Compression.

### 1.3.1 Operating System

MQ Message Compression can be installed on any of the following supported servers:

#### 1.3.1.1 IBM AIX

- IBM AIX 6L version 6.1 or higher

#### 1.3.1.2 HP-UX IA64

- HP-UX v11.23 or higher

#### 1.3.1.3 IBM i (OS/400)

- IBM i V6R1 or higher

#### 1.3.1.4 Linux x86

- Red Hat Enterprise Linux v5, v6, v7, v8
- SUSE Linux Enterprise Server v11, v12, v15

#### 1.3.1.5 Linux x86_64 (64-bit)

- Red Hat Enterprise Linux v5, v6, v7, v8
- SUSE Linux Enterprise Server v11, v12, v15

#### 1.3.1.6 Linux on POWER

- Red Hat Enterprise Linux v6, v7, v8
- SUSE Linux Enterprise Server v12, v15

#### 1.3.1.7 Linux on zSeries (64-bit)

- Red Hat Enterprise Linux v6, v7, v8
- SUSE Linux Enterprise Server v12, v15

#### 1.3.1.8 Raspberry Pi (Linux ARM 32-bit)

- Raspberry Pi OS v9 or higher

#### 1.3.1.9 Sun Solaris

- Solaris SPARC v10 & v11
- Solaris x86_64 v10 & v11

#### 1.3.1.10 Windows

- Windows 2008, 2012 or 2016 Server (32-bit & 64-bit)
- Windows 7, 8, 8.1 or 10 (32-bit & 64-bit)

### 1.3.2 IBM MQ

- IBM MQ v7.1, v7.5, v8.0, v9.0, v9.1 and v9.2 (32-bit and 64-bit)

| Operating System | MQ v7.1, v7.5, v8.0, v9.0, v9.1 and v9.2 |
|---|---|
| AIX v6.1 or higher | 32-bit & 64-bit |
| HP-UX IA64 v11.23 or higher | 32-bit & 64-bit |
| IBM i (OS/400) | 64-bit |
| Linux x86 | 32-bit |
| Linux x86_64 | 32-bit & 64-bit |
| Linux on POWER | 32-bit & 64-bit |
| Linux on zSeries | 32-bit & 64-bit |
| Raspberry Pi ARM | 32-bit |
| Solaris SPARC v10 & v11 | 32-bit & 64-bit |
| Solaris x86_64 v10 & v11 | 32-bit & 64-bit |
| Windows 2008, 2012, 2016, 7, 8, 8.1 & 10 | 32-bit & 64-bit |

### 1.3.3 Windows 32-bit

The following is the software prerequisite for Windows 32-bit:

- Microsoft Visual C++ 2010 Redistributable Package (x86)
  https://www.microsoft.com/en-ca/download/details.aspx?id=5555

### 1.3.4 Windows 64-bit

The following are the software prerequisite for Windows 64-bit:

- Microsoft Visual C++ 2010 Redistributable Package (x64)
  https://www.microsoft.com/en-ca/download/details.aspx?id=14632

If local 32-bit applications connect in bindings mode to the queue manager then the following needs to be also installed:

- Microsoft Visual C++ 2010 Redistributable Package (x86)
  https://www.microsoft.com/en-ca/download/details.aspx?id=5555