WMQ - Tuning Queue Performance on non-z/OS Systems

This updated document makes minor editorial changes.

WMQ Limits

WMQ has many limits built into the system. These are designed to prevent out-of-control applications obtaining important system resources that prevent a balanced workload from operating properly. Customers have stated that one of these limits in particular (Maximum amount of memory (RAM) reserved for non-persistent messages) are hampering normal operations. Therefore, a mechanism for changing these limits has been introduced. However, these limits should be altered only after careful consideration of the potential benefits; change only those queue definitions that require the additional function. This method of altering the limits is, is not an MQSeries API, and hence is not guaranteed to be upward compatible in future releases of the products.

Queue Definitions.

Queues are defined using the DEFINE QLOCAL command, and the information is stored on disk. Any ALTER QLOCAL command changes the information stored on disk. The queue definition consists of information provided by the installation, default values (as specified in the WMQ Command Reference), and limits inserted by the queue manager. One of the standard limit defaults can be changed using the tuning parameters stanza of the QM.INI file. The tuning parameters that affect the queue definition (only) affects definitions created during that invocation of the queue manager.

Maximum amount of memory (RAM) reserved for non-persistent messages

The amount of shared memory (that is, RAM) used to hold non-persistent messages is 64KB per queue on 32 bit Queue Managers (eg Windows) and 128KB per queue on 64 bit Queue Managers(eg UNIX). This can be raised up to 100MB by using the DefaultQBufferSize (see following example). This amount of shared storage is allocated, (together with other control blocks) when the queue is opened and so has a direct impact on the amount of real resources needed by the queue manager. It has a significant effect on both virtual shared memory and real memory. A more realistic buffer size would be less than 1MB and that would depend on the usage of the queue. An iterative method would be to double the 128KB default size(eg UNIX) of the buffer to 256KB and observe the effect on response time. If messages are no longer spilt to disk, they will be removed from the queue more quickly and hence have an

additional effect on the queue depth. Non-persistent messages are initially held in the buffer but subsequent messages spilt out to disk when the buffer fills. The effect is enhanced performance at the expense of storage, but it may degrade the performance of the previously balanced system. If this increase in storage usage pushes the system into paging, the performance will get worse because page faults are more expensive than having MQSeries intelligently manage the disk I/O for non-persistent messages. If there are no persistent messages being processed, then the Persistent buffer will also be used to hold non-persistent messages.

Maximum amount of memory (RAM) reserved for persistent messages

The amount of shared memory (that is, RAM) used to hold persistent messages is 128KB per queue on 32 bit Queue Managers (eg Windows) and 256KB per queue on 64 bit Queue Managers. This can be raised up to 100MB by using the DefaultPQBufferSize. This amount of shared storage is allocated, (together with other control blocks) when the queue is opened and so has a direct impact on the amount of real resources needed by the queue manager. It has a significant effect on both virtual shared memory and real memory. A more realistic buffer size would be less than 1MB and that would depend on the usage of the queue. An iterative method would be to increase the 256KB default size(eg UNIX) of the buffer to 1MB and observe the effect on response time. If messages are in memory, they will be removed from the queue more quickly and hence have an additional effect on the queue depth. Persistent messages are initially held in the buffer (also held on the disk for recovery reasons) but subsequent messages spilt out to disk when the buffer fills.

Opening Queues

The queue definition is stored on DASD and is used to allocate resources at MQOpen. Queue definitions may have been created with various values used for DefaultQBufferSize/DefaultPQBufferSize. Customers who want two queues to have large values and the rest to take defaults, for example will start the queue manager with the specified tuning parameters, create the two queues (definitions stored on disk), and terminate the queue manager. The queue manager is then started without the TuningParameter stanza, and the two queues previously defined with large values keep their large values. Any subsequent queue definitions use the normal defaults. As each queue is opened, the necessary resources are allocated depending on the DASD definition of the queue.

Resource Usage

Customers should test that these changes have not used excessive real resources in their environment and make only those changes that are essential. Allocating several Mega Bytes for several queues reduces the amount of shared virtual storage available for other subsystems, as well as over committing the real storage. When an Empty Queue is Opened, only part of the ‘Buffers’ are allocated, the rest being obtained as additional messages reside on the Queue so any storage analysis needs to be undertaken while the system is messaging.

Here is an example QM.INI file showing the tuning parameters. You should add these tuning stanza lines after the queue manager has been created.

```
#*******************************************************************#
#* Module Name: qm.ini                                             *#
#* Type       : WebSphere MQ queue manager configuration file      *#
#  Function   : Define the configuration of a single queue manager *#
#*                                                                 *#
#*******************************************************************#
#* Notes      :                                                    *#
#* 1) This file defines the configuration of the queue manager     *#
#*                                                                 *#
#*******************************************************************#
ExitPath:
   ExitsDefaultPath=/var/mqm/exits/
#*
#*                                                                 *#
Log:
   LogPrimaryFiles=3
   LogSecondaryFiles=2
   LogFilePages=1024
   LogType=CIRCULAR
   LogBufferPages=0
   LogPath=/var/mqm/log/tstqm/
   LogWriteIntegrity=TripleWrite
Service:
   Name=AuthorizationService
   EntryPoints=10
ServiceComponent:
   Service=AuthorizationService
   Name=MQSeries.UNIX.auth.service
   Module=/opt/mqm/lib/amqzfu
   ComponentDataSize=0
TuningParameters:
   DefaultQBufferSize=1024000
   DefaultPQBufferSize=512000
```

There are no error messages or other feedback to acknowledge that these tuning parameters have been seen, recognized, or used. If they are used to set values larger than permitted, they have no effect.

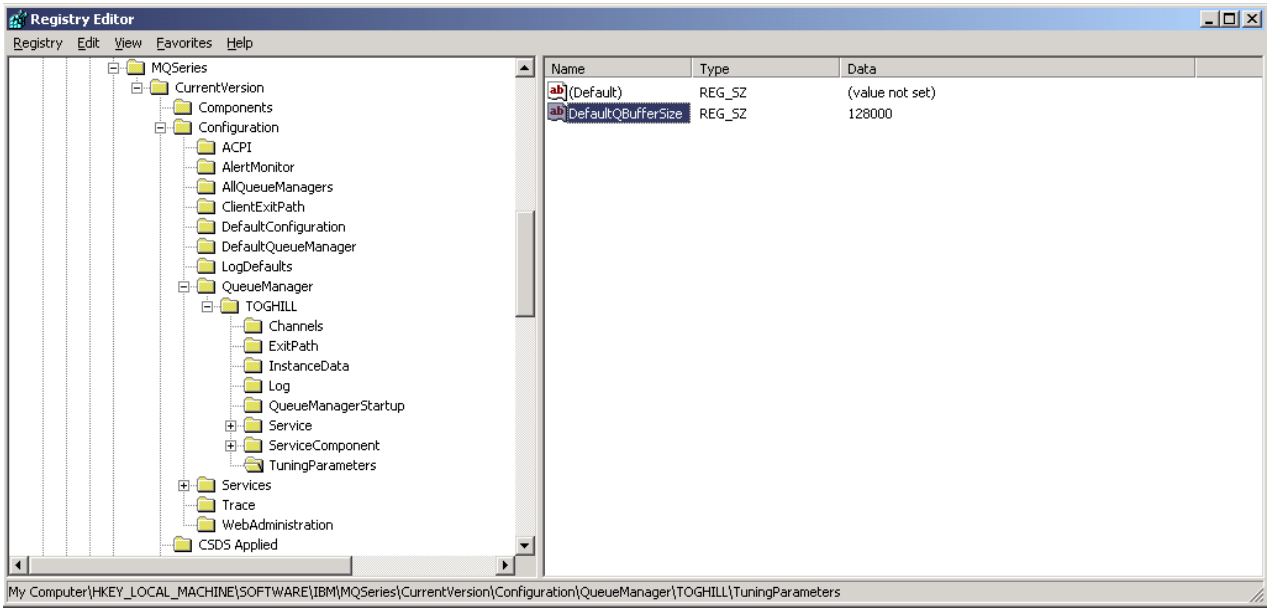
It is easy to degrade the system performance by inappropriate use of these tuning parameters.

Windows

Windows does not use the QM.INI but holds the information in the registry. TuningParameters can be added by using

amqmdain –r TOGHILL -c add –s TuningParameters –v DefaultQBufferSize=128000. (or value less than 100MB)

The Queue manager may need to be started with STRMQM rather than with the Graphical panels.


Registry Editor
Registry Edit View Favorites Help
MQSeries
CurrentVersion
Components
Configuration
ACPI
AlertMonitor
AllQueueManagers
ClientExitPath
DefaultConfiguration
DefaultQueueManager
LogDefaults
QueueManager
TOGHILL
Channels
ExitPath
InstanceData
Log
QueueManagerStartup
Service
ServiceComponent
TuningParameters
Services
Trace
WebAdministration
CSDS Applied
Name
Type
Data
(Default)
REG_SZ
(value not set)
DefaultQBufferSize
REG_SZ
128000
My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\IBM\MQSeries\CurrentVersion\Configuration\QueueManager\TOGHILL\TuningParameters