Configuring and tuning WebSphere MQ for performance on Windows and UNIX

The default configuration for a WebSphere MQ queue manager functions well with average processing loads, but is not optimised for performance. This article shows you how to optimize message processing performance for a WebSphere MQ queue manager running on Windows, UNIX, or Linux.

Introduction

An IBM® WebSphere® MQ queue manager that is created with default properties is configured to produce a fully functioning queue manager using reasonable amounts of memory and disk space. But it is not optimised for performance, and you can make a number of configuration changes to improve the performance of message processing with WebSphere MQ. This article shows you how to make those optimisations for a WebSphere MQ queue manager running on Windows®, UNIX®, or Linux®. The tuning options cover:

- Queue manager log
- Queue manager channels
- Queue manager listeners
- Queue buffer sizes

This table shows which areas of tuning apply to which message types:

	Applies to non-persistent messages	Applies to Persistent Messages
Queue manager log	N	Y
Queue manager channels	Y	Y
Queue manager listeners	Y	Y
Queue buffer sizes	Y	Y

Some of these tuning changes for the queue manger log must be implemented *before* a queue manager is defined, so read thoroughly before doing any set-up or you may have to repeat you work. Such changes are identified in the relevant sections.

Recommendation: Apply tuning to all connected queue managers, because messaging performance using more than one queue manager will depend on the performance of those other queue managers.

You should have some experience with the configuration of WebSphere MQ on Windows and UNIX. In this article, descriptions of parameters and their values are based on WebSphere MQ V6, and a queue manager name of MyQueueManager is used. Substitute the name of your queue manager in registry key names and directory names where appropriate.

Configuration of WebSphere MQ on UNIX and Linux uses the same approach of a qm.ini configuration file. All references to configuration parameters on UNIX apply to Linux as well, although only UNIX is mentioned.

Queue manager log

Queue manager log configuration and performance is only an issue when persistent messages are being processed within the queue manager. If you are using non-persistent messages only, you can skip this section.

Important performance factors include:

- Log file location
- Level of log write
- Type of logging
- Log file size
- Number of log file extents
- Log buffer size
- Number of concurrent applications
- Application processing within a unit of work

Of these, log file location, type of logging, and log file size *must* be specified before a queue manager is created as options on the crtmqm command if you want to change the default values. The values cannot be changed after the queue manager has been created, so you will need to delete and redefine the queue manager to make changes. You can change the other items after the queue manager has been created by specifying the new values in the Log stanza of the queue manager in the Windows registry or in the qm.ini configuration file.

The number of concurrent applications and the application processing within a unit of work are not really WebSphere MQ configuration issues, but depend instead on the applications that access messages in the queue manager. The behaviour of these applications can have a noticeable impact on queue manager performance, especially when something goes wrong.

Log file location

Overview

Recommendation: Locate the queue manager log on its own disk, particularly when you intend to process large messages or high message volumes (> 50 messages per second). When possible, allocate the log on a device with a battery-backed write cache. Such devices are now common in Storage Area Networks (SANs). If that is not practical, use the fastest local disk available -- for example, it is better to use a 10,000 RPM disk than a 6,000 RPM disk.

The speed of the device on which the queue files are located is not so critical to performance. The queue manager uses lazy writes to the queues but synchronous writes to the log, so if you only have one high-performance disk, then allocate it to the log.

The setup of the log file is different on Windows compared to UNIX. However, the way you specify the location is the same for both environments -- you use the -ld option of the crtmqm command.

If you are allocating specific disks for the queue file and log data these must be defined before the queue manager is defined.

Set-up on Windows

Create a directory on the best device available using Windows facilities. For example, create a directory called D:\MQM_LOG\ for the log. If you have multiple queue managers on one operating system image, use different disks for each log.

Specify the directory using the -ld flag in the `crtmqm` command when creating the queue manager. Here is the format of the crtmqm command:

```
crtmqm [-z] [-q] [-c Text] [-d DefXmitQ] [-h MaxHandles]
[-g ApplicationGroup]
[-t TrigInt] [-u DeadQ] [-x MaxUMsgs] [-lp LogPri] [-ls LogSec]
[-lc | -ll] [-lf LogFileSize] [-ld LogPath] QMgrName
```

Set-up on UNIX

On UNIX, you need to allocate a file system to house the queue manager files and log.

Recommendation: Create different file systems on different disks for the queue manager queue and log files. Use the facilities of the operating systems to allocate the

file systems. Allocate the file system for the log on the best possible device available. Here is an example of this procedure on UNIX (AIX):

```
Filesystem    512-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd4          524288    483008    8%     2284     5% /
/dev/hd2         3932160    111160   98%    40644    72% /usr
/dev/hd9var      4456448   2067624   54%    13534     6% /var
/dev/hd3         2097152   1642944   22%      741     1% /tmp
/dev/hd1          262144    261336    1%       16     1% /home
/proc                  -         -    -         -     -  /proc
/dev/hd10opt     9437184         0  100%    12849    47% /opt
/dev/mqmlv      16777216  11453824   32%      561     1% /var/mqm
/dev/mqmloglv   16777216  15725288    7%        9     1% /var/mqm/log
/dev/mqmerrlv   16777216  16552328    2%       82     1% /var/mqm/errors
/dev/db2lv      16777216  15973240    5%      352     1% /db2data
/dev/db2loglv   16777216  16774000    1%        4     1% /db2log
```

The file system allocated on the mqmlv logical volume is mounted as /var/mqm, giving a different file system from /var for WebSphere MQ.

Similarly, the file system allocated on the mqmloglv logical file system is mounted as /var/mqm/log. This file system is allocated so that it resides on its own physical disk.

If the file system /dev/mqmloglv is not mounted as /var/mqm/log file when the queue manager is defined, the log will be allocated in the /var/mqm file system, which is not what you want. Be sure to allocate and mount the file system before running the crtmqm command to create the queue manager.

As an aside, note that the same idea of allocating the data and log has been used for a database installed on the same machine, using the /db2data and /db2log mount points.

Level of log write

Overview

You can specify the method that the queue manager logger uses to reliably write log records. The method used is specified in the Log stanza of the queue manager configuration using the LogWriteIntegrity parameter. Possible values are:

SingleWrite

Some hardware guarantees that if a write operation writes a page and fails for any reason, a subsequent read of the same page into a buffer results in each byte in the buffer being either:

- The same as before the write, or
- The byte that should have been written in the write operation

On this type of hardware (for example, a SAN write cache enabled disk), it is safe for the logger to write log records in a single write because the hardware assures full write integrity. This method provides the best performance.

DoubleWrite

Default method used in WebSphere MQ V5.2 and available for back-compatibility purposes only.

TripleWrite

Default method. When hardware that assures write integrity is not available, you should write log records using the TripleWrite method because it provides full write integrity. Systems running with high volumes (> 1000 messages per second) will see little difference between SingleWrite and TripleWrite, because it is only the last 4k block in each log write that may be subjected to three writes.

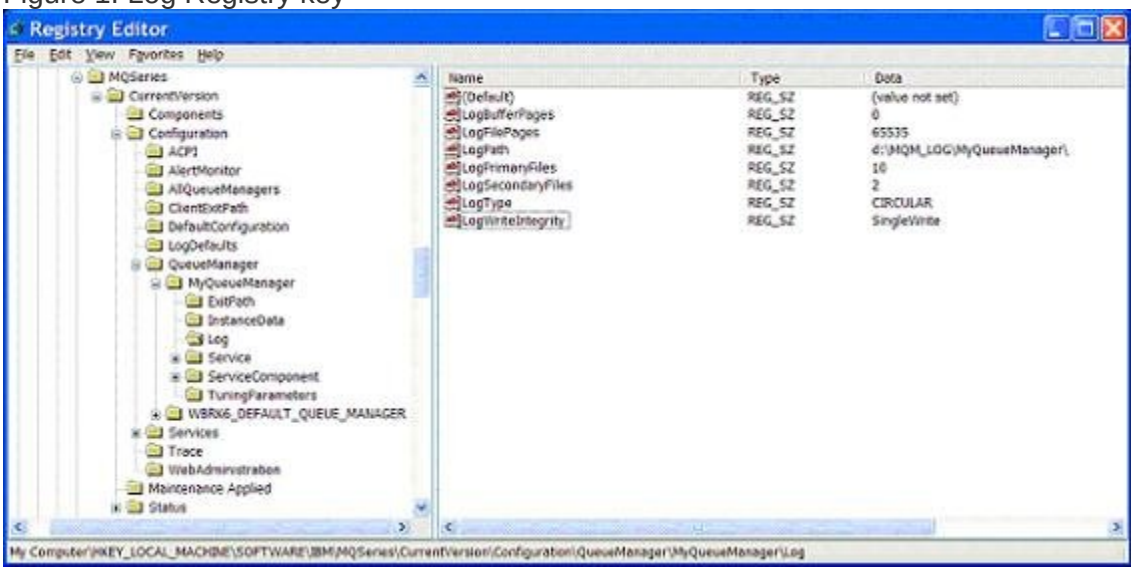
If you are satisfied as the result of a discussion with your disk provider that the device on which one which the log is located can assure write integrity, then use SingleWrite for best performance. If you change the value, you must restart the queue manager to bring the change into effect.

Changing on Windows

To change the level of log write used for the queue manager, you must change the default value in the registry:

1. Stop the queue manager and any associated applications.
2. Back up the Windows registry.
3. Run regedit.
4. Navigate to the queue manager log registry entry. For example: HKEY_LOCAL_MACHINE\SOFTWARE\IBM\MQSeries\CurrentVersion\Configuration\QueueManager\MyQueueManager.
5. Select the Log key.
6. Double-click on the LogWriteIntegrity string value and modify the value as required.
7. Exit regedit.
8. Restart the queue manager. Figure 1 below shows the Log key values for the queue manager MyQueueManager:

Figure 1. Log Registry key



Changing on UNIX

To change the level of log write used for the queue manager, you must change the default value in the queue manager qm.ini file:

1. Stop the queue manager and any associated applications.
2. Back up the file /var/mqm/qmgrs/MyQueueManager/qm.ini.
3. Edit the file /var/mqm/qmgrs/MyQueueManager/qm.ini.
4. Edit the Log stanza if one already exists if not create one.
5. Modify the entry for LogWriteIntegrity to the required value.
6. Save the file.
7. Restart the queue manager.

   Here are the Log key values for the queue manager MyQueueManager which is running on AIX. In practice there may be additional stanzas in the qm.ini file:

```
#*******************************************************************#
#* 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/
   ExitsDefaultPath64=/var/mqm/exits64/
#*                                                                 *#
#*                                                                 *#
Log:
         LogPrimaryFiles=10
         LogSecondaryFiles=10
         LogFilePages=1024
         LogType=CIRCULAR
         LogBufferPages=0
         LogPath=/var/mqm/log/MyQueueManager/
         LogWriteIntegrity=SingleWrite
```

Type of logging

Overview

WebSphere MQ provides two types of logging which are known as circular and linear. When using circular logging log file extents are reused once they no longer contain active log data. When using linear logging on the other hand log file extents will be continually allocated as required. Once a log is no longer used it then becomes available to be archived.

If you need to be able to forward recover queue data following a failure or recover from media failure of the device containing the log you must use linear logging if you are dependent on WebSphere MQ to provide this level of protection for you. An alternative strategy is to use disk mirroring to mirror the log device. This is often a facility provided by a SAN. In this case you could use circular logging.

For performance choose circular logging. Circular logging is the default option when creating a queue manager.

The same considerations apply to both the Windows and UNIX environments when specifying the type of logging.

Use the -lc option on the crtmqm command to specify circular logging, and the -ll option to specify linear logging. Although the type of logging can be specified in the qm.ini file of the queue manager, changes made there will not result in a change in behaviour, as the type of logging cannot be changed once a queue manager has been created.

Log file extent size

Overview

The size of each log file extent is specified on queue manager creation and cannot be changed subsequently so it is important to get this right when first defining the queue manager.

The log file size is specified in the same way for both Windows and UNIX. This is using the -lf parameter of the crtmqm command. There is some difference in the default values though for the two platform types.

In WebSphere MQ for Windows, the default number of log file pages is 256, giving a log file size of 1 MB. The minimum number of log file pages is 32 and the maximum is 65 535.

In WebSphere MQ for UNIX systems, the default number of log file pages is 1024, giving a log file size of 4 MB. The minimum number of log file pages is 64 and the maximum is 65 535.

As long as you have the disk space you are recommended to allocate the maximum size.

Number of log file extents

Overview

Log file extents can be specified as primary or secondary. Primary extents are allocated and formatted by the queue manager when it is first started or when extra extents are added. Once a primary extent has been formatted it can be reused. Secondary log file extents are allocated dynamically by the queue manager when the primary files are exhausted. As these are formatted dynamically it is not recommended that you get to the point where you need them. However they are extremely useful if there is an unexpected peak in activity which results in all of the primary extents being filled (due to a long running unit of work for example). If the primary extents do fill and there are no more secondary extents available the queue manager will resort to backing out uncommitted Units of Work. Given this behaviour ensure that there are a reasonable number of secondary extents.

The number of log extents can be specified on the crtmqm command using the -lp flag for primary extents and -ls flag for secondary extents or by using the LogPrimaryFiles and LogSecondaryFiles values in the Log stanza of the queue manager. For Windows the Log stanza entry of a queue manager is located in the Windows registry. On UNIX the Log stanza is located in the qm.ini configuration file of the queue manger.

The minimum number of primary log files you can have is 2 and the maximum is 254 on Windows, or 510 on UNIX systems. The default is 3.

The total number of primary and secondary log files must not exceed 255 on Windows, or 511 on UNIX systems, and must not be less than 3.

Operating system limits can reduce the maximum possible log size.

The number of extents needed depends on the amount of data to be logged and the size of each extent.

A practical starting point could be values of LogPrimaryFiles=10 and LogSecondaryFiles=10.

Changing on Windows

If you wish to change the number of log file extents that are allocated for the queue manager you will need to change the values in the registry:

1. Stop the queue manager and any associated applications.
2. Back up the Windows registry.
3. Run regedit.
4. Navigate to the queue manager log registry entry. For example HKEY_LOCAL_MACHINE\SOFTWARE\IBM\MQSeries\CurrentVersion\Configuration\QueueManager\MyQueueManager.
5. Select the Log key.
6. Double click on the LogPrimaryFiles string value and modify the value as required to change the number of primary extents.
7. Do a similar thing with the key LogSecondaryFiles if you wish to change the number of secondary extents.
8. Exit regedit.
9. Restart the queue manager.

See Figure 1 above for a view of the LogPrimaryFiles and LogSecondaryFiles parameters in the Log key.

Changing on UNIX

If you wish to change the number of log file extents that are allocated for the queue manager you will need to change the default value in the queue manager qm.ini file:

1. Stop the queue manager and any associated applications.
2. Back up the file /var/mqm/qmgrs/MyQueueManager/qm.ini.
3. Edit the file /var/mqm/qmgrs/MyQueueManager/qm.ini.
4. Edit the Log stanza if one already exists if not create one.

5. Modify the entry for LogPrimaryFiles to the required value to change the number of primary extents. Do a similar thing with the key LogSecondaryFiles if you wish to change the number of secondary extents.
6. Save the file.
7. Restart the queue manager.

For a view of the LogPrimaryFiles and LogSecondaryFiles parameters in the Log stanza of the qm.ini file, see "Changing on Unix" under Level of log write above.

Log buffer size

Overview

The Log Buffer is the amount of main memory used to accumulate log records that will be written out to disk. Log records are appended to the end of the used part of the log buffer. When the end of the buffer is reached, some serialisation takes place that reduces the rate of data transfer to disk. A large buffer will not hit this limit as frequently as a smaller buffer.

You can specify the size of the buffer in units of 4 KB pages using the LogBufferPages parameter of the Log stanza of the queue manager dependent on the platform on which the queue manager is located.

The minimum number of buffer pages is 18 and the maximum is 4096. Larger buffers lead to higher throughput, especially for larger messages.

If you specify 0 (the default), the queue manager selects the size. In WebSphere MQ Version 6.0 this is 128 (512 KB).

If you specify a number between 1 and 17, the queue manager defaults to 18 (72 KB). If you specify a number between 18 and 4096, the queue manager uses the number specified to set the memory allocated.

The value is specified in the registry on Windows and in the qm.ini file on UNIX. It can be changed after a queue manager has been defined. However, a change in the value is not effective until the queue manager is restarted.

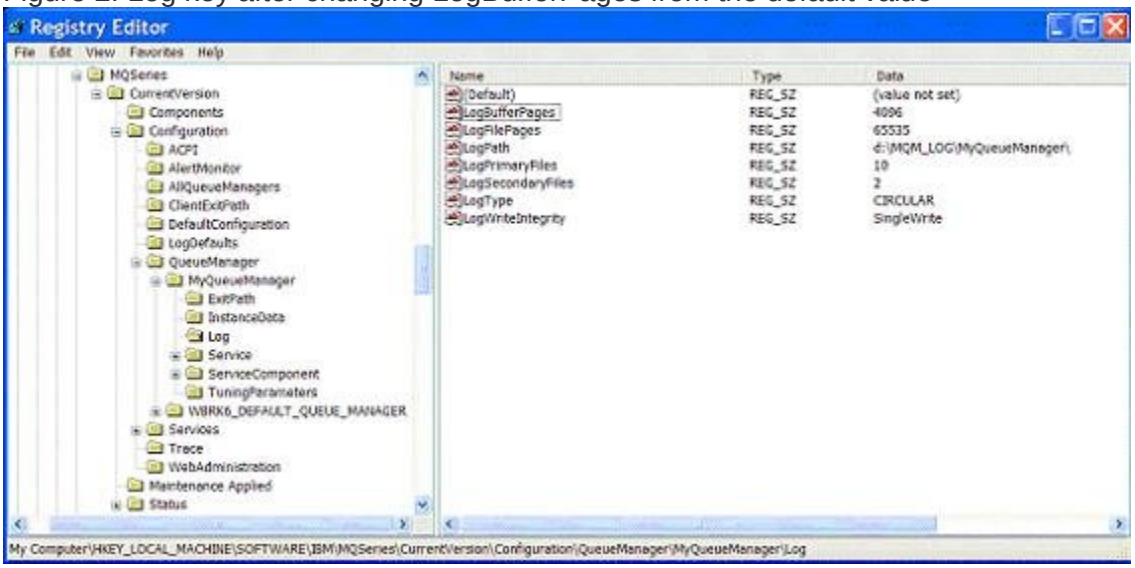
For performance specify the largest possible value. This will help writes of large amounts of log data and enable large messages to be written in a single log I/O. This will be at the cost of some increased storage usage. Using a large value will not hurt performance when writing small amounts of data, but if a small value is specified when writing large amounts of data it can result in the queue manager having to issue multiple writes to the log so impacting performance.

Changing on Windows

If you wish to change the size of the log buffer for the queue manager you will need to change the value in the registry:

1. Stop the queue manager and any associated applications.
2. Back up the Windows registry.
3. Run regedit.
4. Navigate to the queue manager log registry entry. For example HKEY_LOCAL_MACHINE\SOFTWARE\IBM\MQSeries\CurrentVersion\Configuration\QueueManager\MyQueueManager.
5. Select the Log key.
6. Double click on the LogBufferPages key and modify the value as required to change the number of pages allocated.
7. Exit regedit.
8. Restart the queue manager. Figure 2 below shows the Log key after changing LogBufferPages to a value of 4096.

Figure 2. Log key after changing LogBufferPages from the default value



Changing on UNIX

If you wish to change the size of the log buffer for the queue manager you will need to change the value in the queue manager qm.ini file:

1. Stop the queue manager and any associated applications.
2. Back up the file /var/mqm/qmgrs/MyQueueManager/qm.ini .
3. Edit the file /var/mqm/qmgrs/MyQueueManager/qm.ini.
4. Edit the Log stanza if one already exists if not create one.

5. Modify the entry for LogBufferPages to the required value to specify the number of pages to be allocated.
6. Save the file.
7. Restart the queue manager.
   Here is the Log stanza after changing LogBufferPages to a value of 4096. In practice there may be additional stanzas in the qm.ini file.

```
#*******************************************************************#
#* 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/
   ExitsDefaultPath64=/var/mqm/exits64/
#*                                                                 *#
#*                                                                 *#
Log:
         LogPrimaryFiles=10
         LogSecondaryFiles=10
         LogFilePages=4096
         LogType=CIRCULAR
         LogBufferPages=4096
         LogPath=/var/mqm/log/MyQueueManager/
         LogWriteIntegrity=SingleWrite
```

Number of concurrent applications

When processing persistent messages it is recommended to run many instances of the application concurrently in order to optimise the efficiency of the queue manager log. It is possible to have tens or hundreds of applications running concurrently without any detrimental impact on queue manager log performance. In this situation the overhead of a log write at commit time is shared amongst many applications.

Application processing and units of work

When processing persistent messages in an application you should ensure that all MQPUT and MQGET activity takes place within a unit of work, or syncpoint as it is sometime referred to, for efficiency purposes.

Every MQPUT, MQGET, MQCMIT of a persistent message causes a log record to be created for recovery purposes. These records must be successfully written to the queue manager log for the application to correctly commit its updates. The timing and frequency of writes to the log will differ dependent on whether messages are processed in a unit of work or not.

When applications issue all of their MQPUT and MGET operations within a unit of work the queue manager is able to allow multiple applications to concurrently process different messages on the same queue. When these applications issue a commit (MQCMIT) a shared lock can be taken against the queue. This means multiple applications can commit updates at the same time. This is good for message throughput. In this case all of the log records created during the lifetime of unit of work have to be successfully written to the log in order for the commit to complete successfully. This forcing of the log records only has to take place once per unit of work for each application.

Applications which process persistent messages outside of a unit of work or syncpoint have to wait after each and every MQGET or MQPUT for the log record to be synchronously written to the log. This can be many times per application invocation. As individual I/O times can vary from half a millisecond when using a disk with a write Cache to 5-10 milliseconds for SCSI disks this can make a significant difference to overall performance. In addition, while that log write is taking place access to the queue is inhibited for other applications. This disrupts overall processing and is poor for message throughput.

From a performance point of view this is why it is important that ALL applications process persistent messages within a unit of work. It takes only a single application to not adhere to the rule to have a detrimental impact on the performance of persistent message processing.

However, being practical, there may be some situations where you want to write a message outside of the unit of work so that you are sure the write will take place regardless of the success or failure of the unit of work. This is sometimes used for audit purposes where we want to track a particular activity. In such cases functional requirements may override performance considerations.

Queue manager channels

The channels over which a WebSphere MQ queue manager receives and send messages to/from other queue managers can be run as trusted or Fastpath MQ applications.

The attraction of running an MQ application as trusted is that it gives a performance benefit through reduced code pathlength. Fastpath applications run in the same process as the parts of the queue manager so making communication with the queue manager more efficient. This means that there is no process separation between the Fastpath application and the queue manager. By contrast, when running as a non Fastpath or standard application an agent process called amqzlaa provides the separation between the MQ application and queue manager. In this case there is greater separation between the application and queue manager but at the cost of a performance overhead. As channels are WebSphere MQ product code and as such are stable you can freely run channels as trusted applications.

There are a couple of considerations to bear in mind:

- If channel exits are used you may wish to reconsider running the channel as a trusted application as there is the potential for the exit to corrupt the queue manager if the exits are not correctly written and thoroughly tested.
- If you use the command STOP CHANNEL(TERMINATE) you should also reconsider running the channels as trusted applications.
- If your environment is unstable with regular component failure you may also wish to reconsider running the channels as trusted applications.

To make the channels run as trusted there are two options.

1. Specify a value of MQIBindType=FASTPATH in the Channels stanza of the qm.ini or registry file. This is case sensitive. If you specify a value that is not valid it is ignored. See below for how to do this for the Windows and UNIX environments. By choosing this option all channels within the queue manager will run as trusted.
2. Set the environment variable MQ_CONNECT_TYPE to a value of FASTPATH in the environment in which the channel is started. Ensure that the setting MQ_CONNECT_TYPE=FASTPATH is present as an environment variable. This is case sensitive. If you specify a value that is not valid it is ignored.

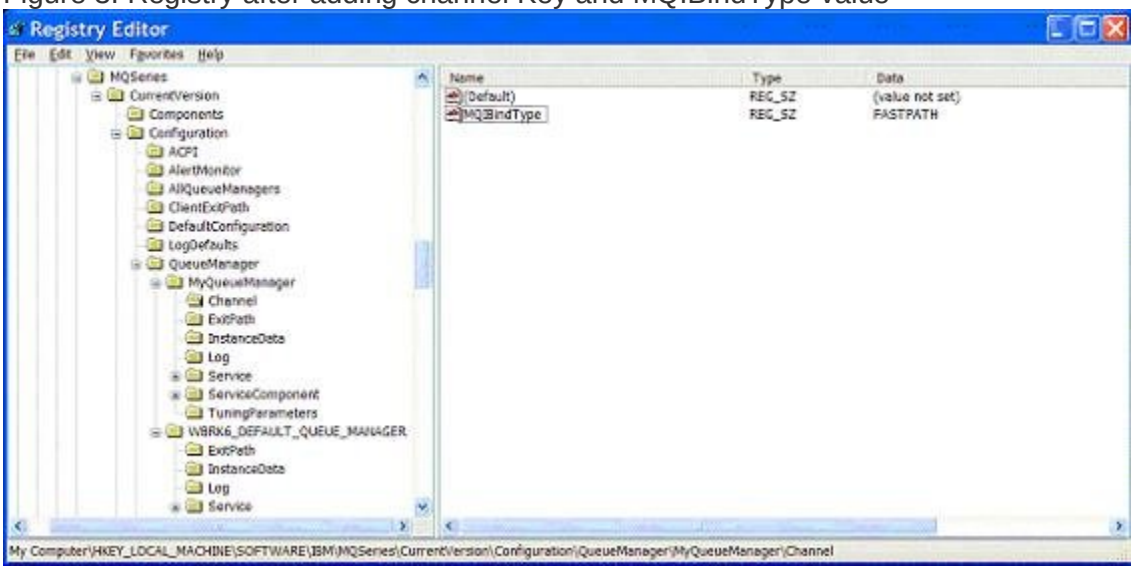
You are recommended to use only one of the options.

Changing on Windows

If you wish to run channels as Fastpath or trusted applications for the queue manager you will need to add a parameter to the registry:

1. Stop the queue manager and any associated applications.
2. Back up the Windows registry.
3. Run regedit.
4. Navigate to the queue manager log registry entry. For example HKEY_LOCAL_MACHINE\SOFTWARE\IBM\MQSeries\CurrentVersion\Configuration\QueueManager\MyQueueManager.
5. Create a Channel key if one does not already exist.
6. Within the Channel key create a string value called MQIBindType.
7. Set the value of MQIBindType to be FASTPATH.
8. Exit regedit.
9. Restart the queue manager Figure 3 below shows the channel key for a queue manager running on Windows where the MQIBindType value has been added.

Figure 3. Registry after adding channel Key and MQIBindType value



Changing on UNIX

If you wish to run channels as Fastpath or trusted applications for the queue manager you will need to change the value in the queue manager qm.ini file.

1. Stop the queue manager and any associated applications.
2. Back up the file /var/mqm/qmgrs/MyQueueManager/qm.ini .
3. Edit the file /var/mqm/qmgrs/MyQueueManager/qm.ini.
4. Add a Channel stanza if one does not already exist.

5. Within the Channel stanza add the MQIBindType value and assign a value of FASTPATH.
6. Save the file.
7. Restart the queue manager.
   Here is the Channels stanza after adding the MQIBindType value. In practice there may be additional stanzas in the qm.ini file.

```
#*******************************************************************#
#* 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     *#
#*                                                                 *#
#*******************************************************************#
CHANNELS:
  MQIBindType=FASTPATH
```

Queue manager listeners

In the same way that channels can be run as trusted applications you can also run a listener as a trusted application.
To do this, set the environment variable MQ_CONNECT_TYPE to a value of FASTPATH in the environment in which the listener is started, in order to ensure that MQ_CONNECT_TYPE=FASTPATH is present as an environment variable. This is case sensitive. If you specify a value that is not valid, it is ignored. This method will work for listeners which:

1. Are started when the runmqlsr command is run manually or in a script.
2. Have been defined using the `Define listener` command in runmqsc.

With option 1, MQ_CONNECT_TYPE=FASTPATH only needs to be present in the shell from which the runmqlsr command is issued. It does not need to be present for other parts of the queue manager.
With option 2, MQ_CONNECT_TYPE=FASTPATH must be set in the environment in which the queue manager is started. This might typically be achieved by setting the environment variable in the .profile of the userid under which the queue manager runs. This is the same for both the Windows and UNIX environments.

Agent processes

If you have decided to use trusted applications throughout one useful check to ensure that there are no agent processes is to run the command `ps -ef |grep amqzlaa.` There should be none. It is difficult to identify which MQ application a particular agent is associated with. If you need to remove an agent you will have to do it by a process of elimination.
As an aside a useful command to look at the environment variables of a process is the command `ps eww.` This is run with a process number as an additional argument.
So run `ps eww 106896` for example. This command can be used when looking to see if MQ_CONNECT_TYPE is set for an MQ listener for example.

Queue buffer sizes

Each queue in the queue manager is assigned two buffers to hold messages, one for non-persistent messages and one for persistent messages. After messages spill out of the buffer, they move to the operating system file system. You can change the buffer sizes to limit this overspill so that data is more readily available to the queue manager. The buffer for the non-persistent messages has a default size of 64K for the 32-bit queue managers (Windows, Linux32) and 128K for 64-bit Queue Managers (AIX, Solaris, HPUX, Linux64). The buffer for persistent messages has a default size of 128K for 32-bit Queue Managers and 256K for 64-bit Queue Managers. The maximum size supported for both queue buffers is 100MB.
The impact of changing the buffer sizes is to increase storage requirements for the queue manager. The increase in storage depends on the size of the buffers and the number of queues to which it applies.
Defining queues using large non-persistent or persistent queue buffers can degrade performance if the system is short of real memory, either because a large number of queues have already been defined with large buffers, or for other reasons -- for example, a large number of channels being defined. Proceed cautiously in increasing these values, and verify that increasing the size improves message throughput. If this is not the case, then do not continue to increase the buffer sizes.
When choosing values, you need to determine the average size and average number of messages that will be on the queue. As with all queue-based processing in WebSphere MQ, aim to have low queue depths.

The non-persistent queue buffer size is specified using the tuning parameter DefaultQBufferSize. The persistent queue buffer size is specified using the tuning parameter DefaultPQBufferSize.

Queues can be defined with different values for DefaultQBufferSize and DefaultPQBufferSize. However, you can only specify one single value for DefaultQBufferSize and one for DefaultPQBufferSize at any point in time in the queue manager.

Non-persistent messages

The value used for the non-persistent messages in a queue are those in effect when the queue is defined, which in turn depends on what values are in effect in the queue manager at the time. The values in the queue manager are determined by the values that were present in the TuningParameters stanza when the queue manager was last started.

When the queue manager is restarted with new values, existing queues will keep their earlier definitions and new queues will be created with the current parameter values. When a queue is opened, resources are allocated according to the definition held on disk from when the queue was created. Buffer sizes for queues are retained over queue manager restarts. **To change the buffer sizes:**

1. Delete any existing queues that you want to change.
2. Stop the queue manager.
3. Add or change the TuningParameters stanza in the Windows Registry or qm.ini file.
4. Restart the queue manager.
5. Define the queues.

If you want different queues to have different settings for DefaultQBufferSize, for example, repeat Steps 2, 3, 4, and 5, changing the value of the buffer size in Step 3 and then only defining those queues that need that particular value in Step 5. Then repeat with the next buffer size in the next iteration of Step 3.

The procedure for adding the buffer sizes is different on Windows and UNIX -- both are explained below. There is no feedback about incorrect syntax from the queue manager.

Persistent messages

The value used for the persistent messages in a queue are those in the TuningParameters stanza when the queue manager was started. This is different from the non-persistent buffer, which is tied to the value when the queue was created. When a queue is opened, buffer resources for holding persistent messages are allocated

according to the current Tuning Parameters stanza. To change the persistent buffer size:

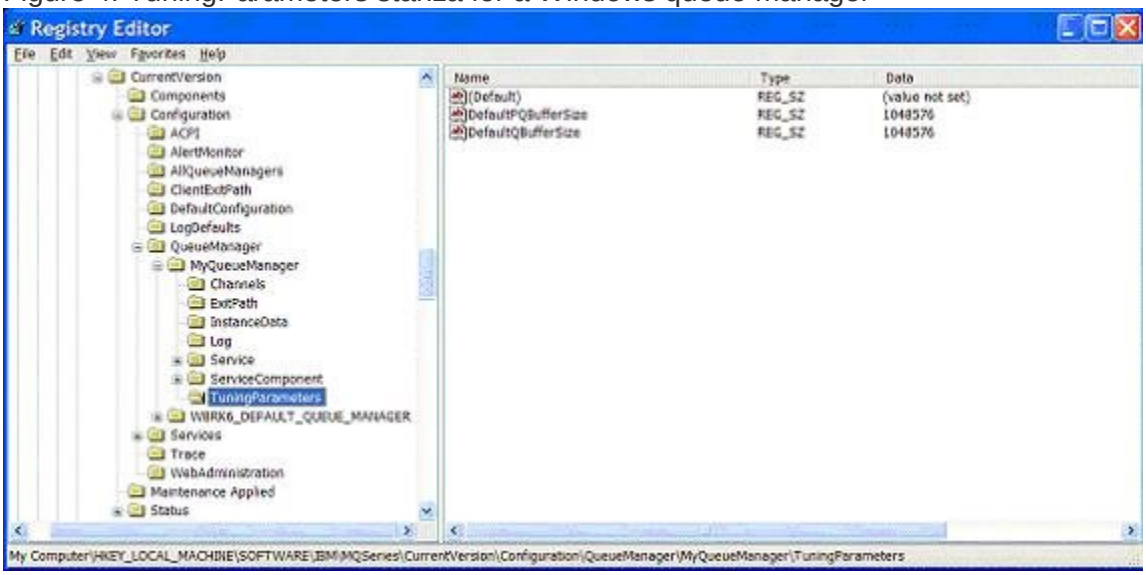
1. Stop the queue manager.
2. Add or change the TuningParameters stanza in the Windows Registry or qm.ini file.
3. Restart the queue manager.

Changing on Windows

1. Delete the queue you want to change.
2. Stop the queue manager and any associated applications.
3. Run regedit.
4. Navigate to queue manager registry entry. For example HKEY_LOCAL_MACHINE\SOFTWARE\IBM\MQSeries\CurrentVersion\Configuration\QueueManager\MyQueueManager.
5. Create a Key called TuningParameters if it does not exist.
6. Navigate to the TuningParameters key.
7. To change the non-persistent buffer size, create a string in the TuningParameters key with the name of DefaultQBufferSize and specify a value in bytes. For example, a value of 1048576 specifies a buffer size of 1 MB. Maximum value is 100MB.
8. To change the persistent buffer size, create a string in the TuningParameters key with the name of DefaultPQBufferSize and specify a value in bytes. For example, a value of 1048576 specifies a buffer size of 1MB. Maximum value is 100MB.
9. In WebSphere MQ V7, you must right-click on the TuningParameters keyword, select permissions, add MQM, and give it full control.
10. Exit regedit.
11. Restart the queue manager.
12. Define the application queues.

Figure 4 shows the TuningParameters key with settings for DefaultPQBufferSize and DefaultQBufferSize equal to 1MB for a V6 queue manager:

Figure 4. TuningParameters stanza for a Windows queue manager



Changing on UNIX

1. Delete the queue you want to change.
2. Stop the queue manager and any associated applications.
3. Backup the file /var/mqm/qmgrs/MyQueueManager/qm.ini .
4. Edit the file /var/mqm/qmgrs/MyQueueManager/qm.ini .
5. If it is not present, add a stanza called TuningParameters.
6. To change the non-persistent buffer size, create an entry in the TuningParameters stanza with the name DefaultQBufferSize and specify a value in bytes. For example, a value of 1048576 specifies a buffer size of 1 MB. Maximum value is 100MB.
7. To change the persistent buffer size, create an entry in the TuningParameters stanza with the name of DefaultPQBufferSize and specify a value in bytes. For example, a value of 1048576 specifies a buffer size of 1MB. Maximum value is 100MB.
8. Save the file.
9. Restart the queue manager.
10. Define the queue(s).

Here is the TuningParameters stanza with settings for DefaultPQBufferSize and DefaultQBufferSize equal to 1MB. In practice, there may be additional stanzas in the qm.ini file.

```
#*******************************************************************#
#* 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     *#
#*                                                                 *#
#*****************************************************************#
TuningParameters:
  DefaultQBufferSize=1048576
  DefaultPQBufferSize=1048576
```

For as long as the DefaultQBufferSize and DefaultPQBufferSize are in place in TuningParameters and the queue manager has been restarted to read those values, then they will apply to all queues that are subsequently defined. If you want to apply the DefaultQBufferSize value only to a specific set of queues, then only define those queues whilst the values are in effect. Remove or change the settings and restart the queue manager before defining any other queues that are to have different values. For further information on changing the queue buffer sizes, seeSupportPac MP01: MQSeries -- Tuning Queue limits in V5 products.

Getting started quickly

This section has a guide to help you quickly configure a queue manager with optimised settings for the processing of persistent and non-persistent messages. This guide is useful for a development or test queue manager. When planning a production environment, take the time to formally plan the amount of log space needed (see the MQ documentation), because running out of log space in a production environment is not an experience you want to go through. Here is the sequence of events:

1. Allocate a separate file system or directory for the log on its own disk. For UNIX the filesystem should be called /var/mqm/log.
2. Ensure /var/mqm/log is mounted before creating the queue manager.
3. Create a queue manager using the command

```
crtmqm -lp 10 -ls 10 -lf 65535 <queue manager_name>
```

. This means:

- Queue manager log file size of 65535 through -lf 65535 option.

- Circular logging is in use. This is the default so no need to specify.
- 10 primary extents and 10 secondary extents for the log specified with the -lp 10 and -ls 10 flags.

4. Save /var/mqm/qmgrs/<queue manager_name>/qm.ini on UNIX or backup the registry on Windows.
5. Edit /var/mqm/qmgrs/<queue manager_name>/qm.ini on UNIX or use regedit on Windows and navigate to HKEY_LOCAL_MACHINE\SOFTWARE\IBM\MQSeries\CurrentVersion\Configuration\QueueManager\<queue manager_name>.
6. Change LogBufferPages in the Log stanza or registry key to a value of 4096 .
7. Create (or add to existing) Channels stanza or registry key and add the following entry, observing case

```
MQIBindType=FASTPATH
```

8. Create (or add to existing) TuningParameters stanza or registry key and add the following entries as values observing case

```
DefaultQBufferSize=1048576
DefaultPQBufferSize=1048576
```

10. Save qm.ini or exit regedit.
11. Ensure MQ_CONNECT_TYPE=FASTPATH is set as an environment variable in the shell or command prompt in which the runmqlsr command is issued.
12. Start the queue manager.
13. Define the application queues.

In this case, the queue buffer sizes have been set to 1MB (1024*1024=1048576). Different values could be used dependent on the requirements.

Why a default queue manager is not optimal

As an illustration of the how the default values for the queue manager differ from the values you might typically want to set when tuning a queue manager, this section describes the default values created for a queue manager on UNIX.

Here is a qm.ini file for a default queue manager on UNIX. The queue manager was created with the command `crtmqm MyQueueManager`, which means that all parameter values default:

```
#*******************************************************************#
#* 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/
   ExitsDefaultPath64=/var/mqm/exits64/
#*                                                                 *#
#*                                                                 *#
Log:
         LogPrimaryFiles=3
         LogSecondaryFiles=2
         LogFilePages=1024
         LogType=CIRCULAR
         LogBufferPages=0
         LogPath=/var/mqm/log/MyQueueManager/
         LogWriteIntegrity=TripleWrite
Service:
         Name=AuthorizationService
         EntryPoints=13
serviceComponent:
         Service=AuthorizationService
         Name=MQSeries.UNIX.auth.service
         Module=/usr/mqm/lib64/amqzfu
         ComponentDataSize=0
```

Whilst this configuration will specify a perfectly functional queue manager capable of processing non-persistent and persistent messages, performance is not optimised:

- LogPrimaryFiles is set to 3 only. We would typically want a higher value.
- LogSecondaryFiles is set to 2 only. We would typically want a higher value.
- LogType is set to circular. This is OK assuming we do not want linear logging.
- LogBufferPages is set to 0, which in practice is a value of 128. This is small if we want to process at a high message rate or process large persistent messages.
- LogWriteIntegrity is set to TripleWrite. Ideally we would want the log on a reliable device and then we could use a value of SingleWrite.
- There is no channel stanza. As such channels will as run as standard applications unless the environment variable MQ_CONNECT_TYPE=FASTPATH is established.
- There is no TuningParameters stanza to allow the value of the default queue buffer size to be changed. As such all queues will have a default buffer size of 64K(32 bit operating system)/128K(64 bit operating system)

for non-persistent message and 128K(32 bit operating system)/256K(64 bit operating system) for persistent messages. Dependent on the size of messages being processed this may be adequate or not.

Similar considerations apply to a default queue manager allocated on a Windows system. However, the value of some specific parameters would be different, as defaults differ between the two operating systems.