Logger enhancements for MQ v9.0.2 and v9.1

MarkWhitlock
Published on 28/08/2018 / *Updated on 29/08/2018*

In MQ v9.0.2, we enhanced linear logging to enable automatic management of log extents and automatic recording of media images, as well as improving the general performance of linear logging scenarios exploiting these capabilities. This addressed some of the most highly voted for RFE's. The decision to choose circular or linear can now be made based primarily upon whether you want media recovery. Now that we've made linear so much easier to administer and brought linear up to a similar performance to circular, these factors cease to be so important in a linear vs circular decision. We also added in some new statistics to help you tune your log, and now attempt to keep the workload in the primaries only. In MQ v9.1 we have also enabled automatic recording of media images on IBM i.

Before v9.0.2, if you used linear logging, you had to manage log extents yourself – deleting them when they were no longer needed for restart or media recovery, typically by using one of the supportpacs or by writing your own logger event listener. Also you had to record media images regularly yourself by calling rcdmqimg periodically.

But with v9.0.2 onwards, the queue manager can automatically manage log extents and automatically record media images, this makes linear logging almost as easy to administer as using circular logging. The performance of linear logging is much improved to the extent that circular logging now only performs slightly better. As it has always done, linear logging also provides media recovery which circular logging does not. By media recovery, I mean it allows you to recover objects and queues if they are damaged.

Log management
From MQ v9.0.2, you can use automatic, archive or manual log management. Manual is the default which behaves exactly the same as before v9.0.2 – so you have to manage log extents yourself.

Automatic log management means the queue manager will perform all the management of log extents itself. When an extent becomes superfluous (so it is no longer needed for restart or media recovery), the queue manager either reuses or deletes it automatically – so there's no need for you to write your own logger event listener anymore. Archive log management is for customers who wish to archive log extents – once your event listener or script has archived an extent, issue SET LOG ARCHIVED to let the queue manager know.

The automatic reuse of log extents is responsible for much of the performance benefit you get with automatic log management. You can recognise extents that are waiting to be reused (we call them reuse extents) because they are prefixed with the letter "R" instead of the usual "S". Switch on log management by passing -lla (for automatic) or -lln (for archive) to crtmqm instead of -ll. To change an existing linear logging queue manager to use automatic or archive log management, set LogManagement=Automatic or LogManagement=Archive in qm.ini and then restart your queue manager.

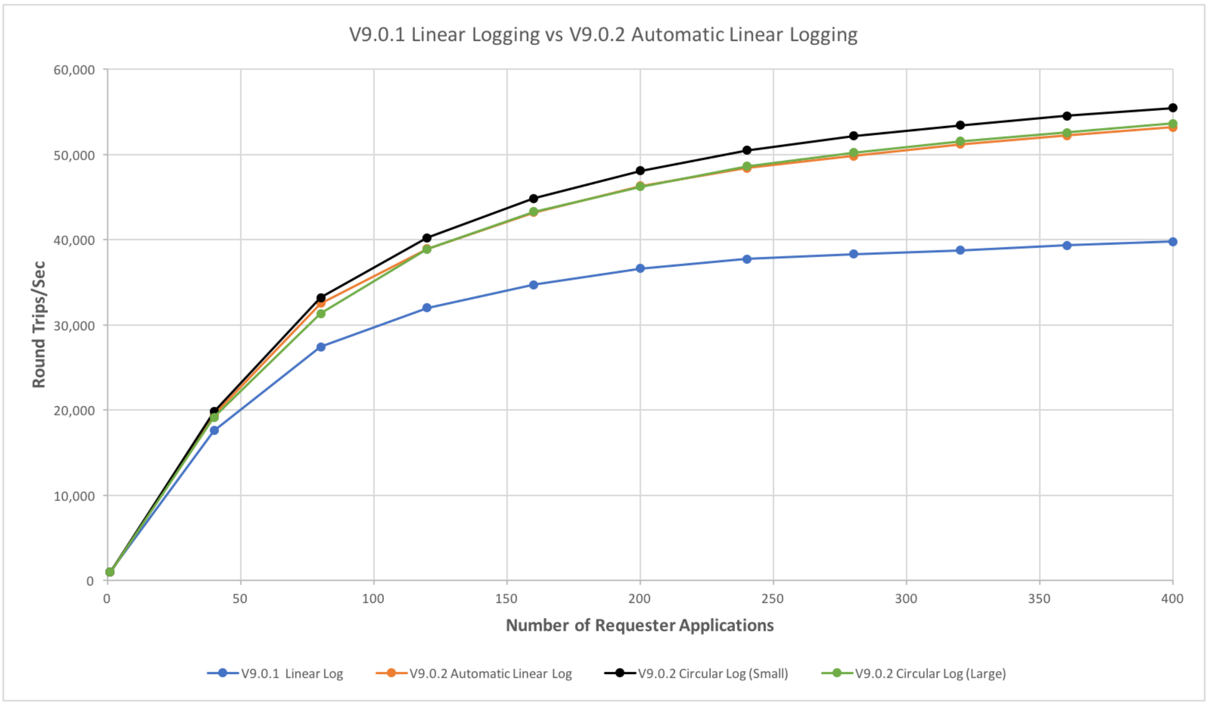

Figure 1

Figure 1 above, shows the improvement in linear logging performance, between V9.0.1 and V9.0.2 (with automatic log management). A workload run that was limited by the rate the queue manager could write to the log files (on SAN storage). The overall round trip rate is shown on the y-axis, which increases as we add more applications putting and getting to the queues. For both linear logging tests, media images were recorded every 10 seconds with rcdmqimg. Whilst the V9.0.1 test will continue to create and format new linear log files continuously, in V9.0.2, existing log files are made available for re-use as the media recovery point is moved forward in the log. Performance is significantly improved in V9.0.2 as a result, coming close the that of circular logging (with an efficiently sized log). Running the same test with an over-sized circular log file set (V9.0.2 Circular Log(Large) in the figure above) shows a similar level of performance to linear logging, indicating that the number of files being used is the main difference between circular and linear logging now, rather than the additional formatting costs.

Automatic recording of media images

Also from v9.0.2, you can ask the queue manager to record media images automatically. Set the new queue manager attribute IMGSCHED(AUTO). The default value is IMGSCHED(MANUAL) which behaves like pre-v9.0.2, i.e. you are expected to call rcdmqimg manually yourself.

Once you have set IMGSCHED(AUTO), also set IMGINTVL and/or IMGLOGLN. IMGINTVL is how often the queue manager will take automatic media images, in minutes. IMGLOGLN is how much log data gets written before the queue manager takes the next automatic media image, in megabytes. Both IMGINTVL and IMGLOGLN are targets – the actual gap between images may be larger or smaller.

Setting IMGLOGLN is often a better idea than only setting IMGINTVL, as setting IMGLOGLN will stop your log getting too large no matter how heavy your workload is. Or you can set both IMGINTVL and IMGLOGLN together and then an image is recorded when either of them expire. In fact setting both can be quite a good idea, so images are taken more frequently during a heavy workload when IMGLOGLN expires, but are still taken occasionally during a light workload when IMGINTVL expires.

You can even record media images manually by calling rcdmqimg when you've enabled automatic recording of media images, if you know it's a really good time to do so, such as a very quiet time. When you do that the queue manager will reset its interval and log length before the next automatic media image is taken.

Beware of setting IMGLOGLN too small though. If you make it a similar sort of size to your largest queue, then the queue manager will spend all its time recording media images, because the media image itself is included in the log data being counted towards IMGLOGLN. Better to make IMGLOGLN a fraction of the size of your log filesystem – maybe a third or less.

There are other performance advantages to automatic recording of media images such as staggering of media images, partial media images, and recording media images early when the queue is empty – but I'll talk more about these later.

Improved documentation
Having problems sizing your log? Not sure how to calculate how many primary and secondary log files you need? Having problems knowing how big to make your log filesystem? There's now much better information in the MQ knowledgecenter to help you with this. Have a look at https://www.ibm.com/support/knowledgecenter/SSFKSJ_9.0.0/com.ibm.mq.con.doc/q018470_.htm and the subpages under it.

Keeping the workload in the primary extents only
The active log is those log extents needed for restart recovery. When the queue manager checkpoints, it is attempting to mark some of the first few active extents as inactive – no longer needed for restart recovery. The active log can only grow to the size of the primary and secondary log extents. That's what primary and secondary log extents mean. Inactive log extents may still be needed for media recovery. When the log is starting to fill (the number of primary/secondary are getting full), the queue manager schedules a checkpoint so extents can be marked as inactive and the active log shrinks.

Before v9.0.2, the queue manager scheduled checkpoints based on the number of primaries plus secondaries – so secondaries were often used for regular workload. From v9.0.2, the queue manager prefers to only use primaries for regular workload. Secondaries should only be used when there are long-running transactions.

This means that the next checkpoint is scheduled when a large fraction of the primaries are being used. If the next checkpoint is scheduled before the previous checkpoint has completed you will get AMQ7466 "There is a problem with the size of the logfile. The log for queue manager is too small to support the current data rate." So if your normal workload has previously been using secondary extents, you may well get this error message in your error logs when you migrate to v9.0.2 or later whether or not you use the new logger enhancements and whether you use linear or circular logging. If you do, increase the number of primary extents, making sure that the total size of all your extents won't exceed the size of your log filesystem, and also making sure that you’re ok with increasing your restart time, since increasing primaries alone increases the maximum size of your active log.

More log attributes and better statistics
Also from v9.0.2, there’re more log attributes returned from runmqsc DISPLAY QMSTATUS LOG that will help you with monitoring your log. As well as the existing attributes such as CURRLOG, RECLOG and MEDIALOG, new attributes returned include ARCHLOG, RECSZ, MEDIASZ, ARCHSZ, REUSESZ, LOGUTIL and LOGINUSE. ARCHLOG is the oldest extent needing archiving (only set for archive log management). ARCHLOG is also passed to your logger event listener, to help you know where to start archiving extents. RECSZ, MEDIASZ, ARCHSZ and REUSESZ are the sizes in megabytes of all the extents needed for each of restart recovery, media recovery, needing archiving and available for reuse.

LOGUTIL and LOGINUSE both give the percentage of the primary files that are being used for the active log – so these statistics can help you better size your log. LOGUTIL and LOGINUSE are different though. LOGINUSE is a point in time statement and so will increase until a checkpoint is taken, when it will typically suddenly decrease as a bunch of extents become inactive. So a rolling average of LOGINUSE would help more than the sawtooth that sampling LOGINUSE would give you. And that’s exactly what LOGUTIL is – a rolling average of the percentage of the primary files that are being used by the active log. So LOGUTIL shouldn’t jump sharply on a checkpoint in the way LOGINUSE does. Of course LOGINUSE and LOGUTIL can both be over 100% – that just means you’re using secondaries. And because the queue manager tries to keep the workload in the primaries, when LOGUTIL goes above 100% it means you have long-running transactions. Perhaps you’re ok with that – but you might want to think about making those transactions shorter, or else increasing your primaries a bit. Best practice is to keep LOGUTIL below 100%.

When an object is damaged
A word of warning though if I have persuaded you that automatic log management and automatic recording of media images means that all the management of log extents is taken care of. If you get a damaged object then you need to act to resolve the problem. An object such as a queue can become damaged if it’s queue file gets deleted or is corrupted.
When using linear logging, you can recover the queue by using rcrmqobj. But until you recover or delete the object, your log filesystem will start filling (just as before V9.0.2). This is because the queue manager will no longer be able to record media images of the damaged object. So the oldest extent needed for the media recovery of the damaged object (and all subsequent ones) will need to be kept by the queue manager. If your workload continues, new log extents will be written to but the old ones can’t be reused until a new image of the object is taken which can’t happen until the damaged object is recovered or deleted – hence the problem.

How long you have until your log filesystem fills up completely depends on how big your log filesystem is and how much log data is being written to disk by your workload. If or when your log filesystem fills, the queue manager will start rolling back transactions to conserve log space. So your workload will get rolled back. While the queue manager should initially stay up, if the situation persists the queue manager may eventually fail.

You will know you have a damaged object because you will get FFDCs as soon as the queue manager realises. It would be prudent to investigate why the object got damaged in the first place, and then recover it quickly which should alleviate the space problems.

Performance advantages of automatic recording of media images
Earlier I mentioned there were more performance advantages with allowing the queue manager to record media images automatically. One such advantage is that these media images will gradually get staggered over time. When a manual image is requested the queue manager is required to respond synchronously to the request and records the image relatively quickly into the log. When an automatic image is scheduled the queue manager does not need to respond synchronously and can write the image over a longer period of time. By writing the image more slowly the act of recording the image itself has less impact on the foreground workload, and gradually results in the images being spread over time. When you first start your queue manager, it may record media images of all objects immediately after one another. But as each object's image is recorded separately, recording these images will gradually get staggered as your workload progresses, meaning that performance hit of recording images is spread out and so is likely to impact your workload less.

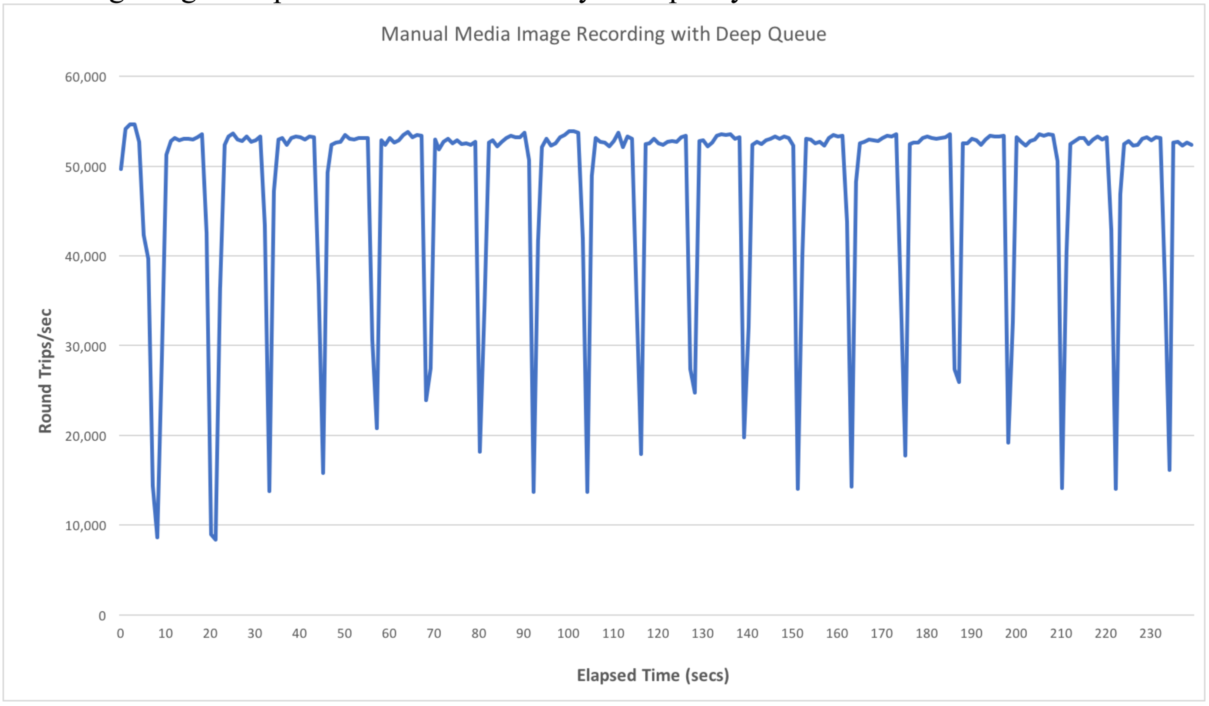

Figure 2

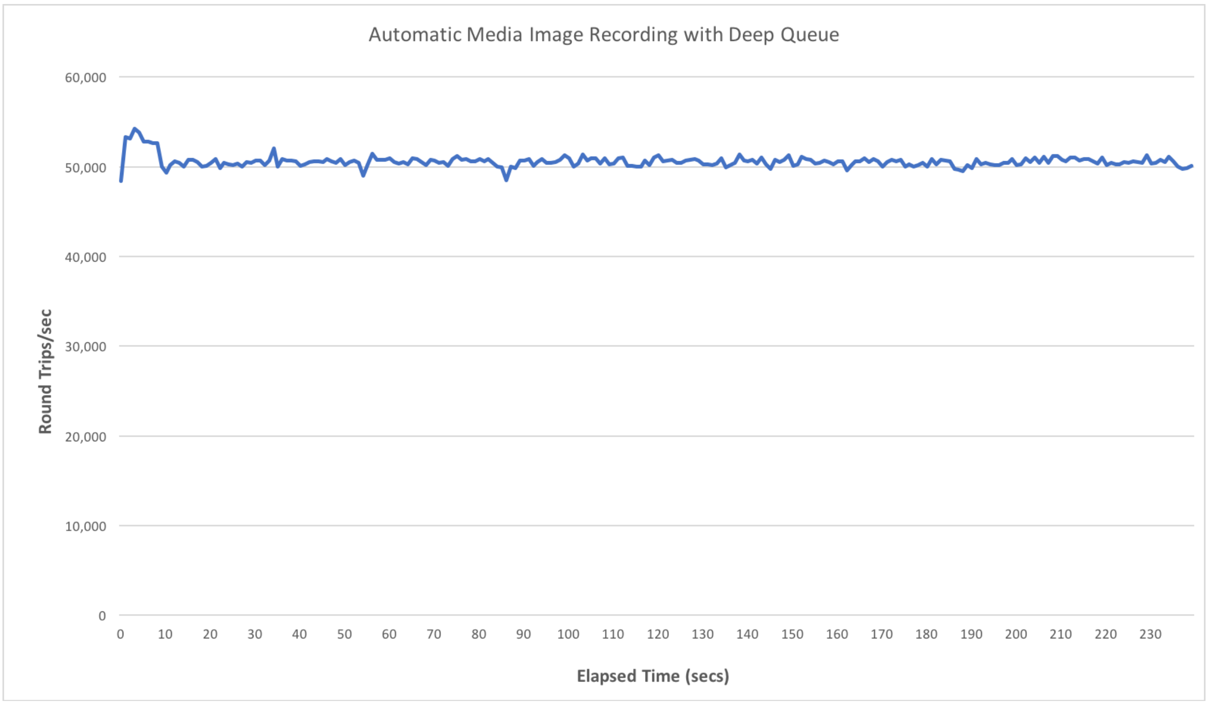

Figure 3

Figures 2&3 above show the difference in impact between manual, and automatic media image recording. A queue manager was configured with a deep queue (1000 x 1MB messages on queue) and a workload run that was limited by the rate the queue manager could write to the log files (round trips/sec on the y-axis). In Figure 2, media images are triggered manually, by running rcdmqimg roughly every 10 seconds (we wait 10 seconds after the completion of each copy). In figure 3, the same test is configured to run with automatic media image recording (with IMGLOGLN set to 400, to approximate the frequency of image recording in the manual test). As you can see in Figure 2, the test is severely impacted by the asynchronous nature of the manual image recording, whereas the automatic media image recording (Figure 3) ameliorates the impact by interleaving the image records with the normal log writes. Since this is a test running at the limits of the log disk bandwidth, the peak rate drops in Figure 3, but if we we had some headroom in terms of log write bandwidth, the media images recorded with the new automatic management might take place with little or no impact on the workload.

Another performance advantage of automatic recording of media images is that the queue manager may be able to record less data into the log. Establishing a point of recovery while not recording a full copy of the object in the log is known as a partial media image. When the queue manager records a partial media image, it does so on behalf of a recovery point which is a little time in the past. If many (or all) of the messages on the queue were put since the recovery point, then these messages do not have to be recorded in the media image, since recovering the queue would replay all log records since the recovery point and so would include all these recent puts. In a similar vein, if many or all of the messages that were on the queue at the recovery point have been got off the queue since, these messages don’t have to be recorded in the partial media image because they aren’t on the queue now. So, if the queue manager is lucky, the partial media image may contain no data at all and so be very quick to record, even though the queue was never empty. This optimization is most effective when most messages only rest on the queue for a short period of time. If you do the opposite and use MQ as a database, using queues as a place to store data long-term, then the queue manager has no alternative but to record full media images every time.

One of the best times to automatically record a media image is when the queue is completely empty, or almost empty, because then there is little user data to log so the media image written is very small. The queue manager looks out for such times and if a media image hasn't been written for a while, but IMGINTVL or IMGLOGLN hasn't expired yet, the queue manager may decide to record a media image anyway because now is a really good time. It is much easier for the queue manager to spot such times than it is likely to be for you, so this provides another performance boost to automatic media image recording.

More runmqsc logger commands
There are a couple of other new runmqsc logger commands from v9.0.2 – RESET QMGR TYPE(REDUCELOG) and RESET QMGR TYPE(ARCHLOG). RESET QMGR TYPE(REDUCELOG) requests the queue manager to reduce the number of reuse extents. RESET QMGR TYPE(ARCHLOG) tells the queue manager you've archived a whole bunch of extents, instead of having to call SET LOG ARCHIVED on each one individually. RESET QMGR TYPE(REDUCELOG) can also be used in a circular logging environment to request the queue manager to try to reduce the number of secondary extents currently allocated.

Also you can make individual queues recoverable or not by setting the new queue attribute IMGRCOVQ or letting it default to the IMGRCOVQ queue manager attribute. You can also make other objects recoverable or not by setting the new queue manager attribute IMGRCOVO. By default when you use linear logging, all objects are recoverable, so you can recover them and media images need to be recorded for them. But if you have objects or queues which contain messages that you don't really care about or could be regenerated or are regularly superceded, then make them not recoverable. This means that media images can't be recorded for them but they can't be recovered either.

IBM i support
In v9.1, some of these new logger enhancements are available on IBM i as well. On IBM i, logging is different because the MQ logger uses journal receivers, which cannot be reused, so only linear logging is supported. In v9.1 on IBM i, automatic recording of media images is supported, but automatic and archive log management is not. So in v9.1 you get all the performance advantages of automatic recording of media images by specifying IMGSCHED(AUTO) along with IMGLOGLN and IMGINTVL.

Conclusion
In summary, the new logger enhancements for v9.0.2 make linear logging much simpler and easier to use, as well as improving performance. And we've also made the documentation in the knowledgecenter quite a lot better https://www.ibm.com/support/knowledgecenter/SSFKSJ_9.0.0/com.ibm.mq.con.doc/q018410_.htm. Most customers who use linear logging pre-v9.0.2 should benefit from moving to automatic log management and automatic recording of media images.

TAGS ARCHLOG, CURRLOG, DAMAGEDOBJECT, IBMI, IMGSCHED, LINEARLOG, LOG, LOGGER, LOGMANAGEMENT, LOGUTIL, MEDIALOG, MQ9, MQ9.0.2, PERFORMANCE, RCDMQIMG, RCRMQOBJ, RECLOG, RECORD MEDIAIMAGE

by MarkWhitlock

4 comments on"Logger enhancements for MQ v9.0.2 and v9.1"

1. **MarkWhitlock** February 13, 2020
   Hi Rak,
   The new log attributes ARCHLOG, RECSZ, MEDIASZ, ARCHSZ, REUSESZ, LOGUTIL and LOGINUSE as well as the existing CURRLOG, RECLOG and MEDIALOG attributes can be used to check how the log is being managed. These attributes are returned from runmqsc DISPLAY QMSTATUS LOG. They enable you to monitor the current extent, and which extents are needed for media and restart recovery, as well as the sizes of all the extents needed for media and restart recovery as well as those needing archiving and available for reuse. For more information, please see the "More log attributes" paragraph above,
   or https://www.ibm.com/support/knowledgecenter/SSFKSJ_9.1.0/com.ibm.mq.ref.adm.doc/q086250_.htm or the subpages
   under https://www.ibm.com/support/knowledgecenter/SSFKSJ_9.1.0/com.ibm.mq.con.doc/q018410_.htm
   Mark
2. **Rakesh** February 07, 2020
   Hi,
   I have configured all the required attributes and could see the log rotation is happening. Is there any way to check the log management information(Like in manual setup). will it writes the output somewhere ?
   Thanks,
   Rak.
3. **MarkWhitlock** April 23, 2019
   Hi Jared,
   Yes, setting IMGINTVL to 360 and IMGLOGLN to 1560 will mean the queue manager automatically records a media image every 6 hours or 1560M, whichever happens first. Don't forget to also set IMGSCHED(AUTO) to enable the queue manager to automatically record media images. When you say your linear log currently has 24 log extents of 65M each, I guess this is the average size of your log, or the target size, and the size of the filesystem that contains the log is much larger. Make sure your log filesystem is much larger than IMGLOGLN otherwise you risk the log filesystem filling.
   Setting IMGSCHED, IMGINTVL and IMGLOGLN automatically records media images, but doesn't manage log extents. To enable automatic management of log extents set LogManagement=Automatic in qm.ini. Automatic log management means the queue manager will automatically reuse log extents that are not needed for restart recovery or media recovery. Automatic log management prevents the log from growing indefinitely as long as media images are being taken regularly.
   I hope this helps,
   Mark
4. **Jared** April 19, 2019
   I am currently on MQ8 and I have cronjobs triggering rcdmqimg and cleanmqlogs every 6 hours, then after 48 hours I delete the zipped logs. I use backups in case I need to go through deleted logs.
   In a transition to MQ9.0.4 I would like to utilize Automatic logging. What settings would you recommend? I have linear logging containing 24 logs of 65MB.
   I have turned on Automatic logging on a test server, but I'm not sure of the parameters to emulate what i have in manual logging.
   Default settings for automatic logging are

IMGLOGLN(OFF) (Image Log Length/Size in MB)
IMGINTVL(60) (Imaging Interval in minutes)
Would I be correct in changing the IMGINTVL to 360 for 6 hours and IMGLOGLN for 24*65?
What other settings should I be looking into?