Can two managed transfers access the same source file at the same time?

Paul_Titheridge
Published on 06/08/2018

I had an interesting question from one of my colleagues in IBM Support last month. He was working with a customer who was using IBM MQ Managed File Transfer (MFT) V9.0.5 to move large files. Due to the size of the files, the managed transfers were taking a while to complete. This got my colleague wondering what would happen if one managed transfer was moving a large file, and then another managed transfer was submitted to the same agent to move the same file. I did some investigation, and this is what I found.

Standard MFT agents will take a shared lock when reading a file, and an exclusive lock when writing files. This results in the following behaviour:

Transferring the same file to different locations

If two or more managed transfers are submitted to an agent to transfer the same file to different locations at the same time, then the managed transfers will run in parallel.

Transferring the same file to the same location, with -de (destination_file_behavior) set to “overwrite”.

If two or more managed transfers are submitted to transfer the same file to the same location, and the managed transfer requests have the -de option set to “overwrite” then the managed transfers will effectively run sequentially.

Although all the managed transfers can access the source file at the same time, only one can access the destination file. This means that the first one that starts will lock the destination file, which blocks all subsequent managed transfers. When the first managed transfer has finished and released the lock on the destination file, the second managed transfer will be able to get it and proceed, and so on.

Here is an example that shows this:

- The source agent for Managed transfer 1 locks source file A for reading.
- The source agent for Managed Transfer 2 also locks source file A for reading.
- The destination agent for Managed Transfer 1 locks destination file B for writing, and starts writing data to it.
- The destination agent for Managed Transfer 2 tries to lock destination file B, and can’t, as it is currently locked for Managed Transfer 1. The destination agent waits for the file to become available.
- The destination agent for Managed Transfer 1 releases the lock on destination file B.

- The destination agent for Managed Transfer 2 gets the lock for destination file B, and starts writing to it.
- The source agent for Managed transfer 1 releases the lock on source file A.
- The destination agent for Managed Transfer 2 releases the lock on destination file B.
- The source agent for Managed Transfer 2 releases its lock on source file A.

Transferring the same file to the same location, with -de (destination_file_behavior) set to “error”.

If two or more managed transfers are submitted to transfer the same file to the same location, and the managed transfer requests have the -de option set to “error”, then the first managed transfer will create the destination file and complete successfully. However, all subsequent managed transfers will fail because the destination file already exists, as shown in the example below:

- The source agent for Managed transfer 1 locks source file A for reading.
- The source agent for Managed Transfer 2 also locks source file A for reading.
- The destination agent for Managed Transfer 1 creates and locks destination file B for writing, and starts writing data to it.
- The destination agent for Managed Transfer 2 tries to create destination file B, and can’t, as the file already exists.
- The source agent for Managed Transfer 2 releases its lock on source file A, and marks the managed transfer as “Failed”.
- The destination agent for Managed Transfer 1 finishes writing data to destination file B, and releases the lock.
- The source agent for Managed transfer 1 releases the lock on source file A.

The effect of the -sd (source_disposition) parameter

One thing to note is that the source disposition for managed transfers, which is processed at the very end of the managed transfer (after any TransferEndExits or post-source/post-destination programs have run), also has an effect here.

If two or more managed transfers that have the source disposition of “delete” are all accessing the same source file at the same time, then the only one that will complete successfully will be the last one – the other managed transfers will all be marked as “Partially successful”, as they will be unable to delete the source file. This is because when the other managed transfers complete and try to delete the source file, there will still be an active managed transfer that has the file locked. For example:

- The first managed transfer starts, and locks source file A for reading.
- The second managed transfer starts, and also locks source file A for reading.
- The first managed transfer completes, and the source agent tries to delete file A. However, it is unable to do so as the file is still locked by the second managed transfer. As a result, the managed transfer is marked as “Partially successful”.
- The second managed transfer completes. The source agent is able to delete file A, as there are no other managed transfers that are accessing it.

Now, the eagle-eyed among you will have spotted that I mentioned “Standard MFT agents” at the start of this blog post. Protocol bridge agents do not lock files when reading from or writing to them, which results in a different set of behaviours. I will cover that in a future blog post!

by Paul_Titheridge

2 comments on"Can two managed transfers access the same source file at the same time?"

1. Anonymous November 07, 2018

   Tangentially related question: Is there any way to specify the location where the partial files (.part) are written? Having to alter processes that monitor directories to exclude those isn’t exactly an elegant solution.

   Reply

   - Paul_Titheridge November 09, 2018

     Hi,

     Thanks for the question. Unfortunately, it is not possible to do this. Destination agents for managed transfers will always write the temporary files (which end with .part) to the same location as the destination file.

     However, using a PostDestinationCall or a DestinationTransferEndExit might help with the scenario you have described.

     It should be possible to submit a managed transfer that writes a file to temporary directory, and then have either a PostDestinationCall or a DestinationTransferEndExit move that file to a different directory. This ensures that any processes which monitor that directory don’t have to exclude .part files, as those files will be written to the temporary directory.

     Hope this helps!

     Paul