File locking in IBM MQ 9.3 multi-instance queue managers

https://www.ibm.com/support/pages/node/608527

Date last updated: 28-Jan-2024

Angel Rivera
IBM MQ Support
https://www.ibm.com/products/mq/support
Find all the support you need for IBM MQ

+++ Objective +++

The objective of this document is to analyze the file locking activity in an IBM MQ 9.3 multi-instance queue manager in Linux for the following files of the queue manager, which help to coordinate when the Standby instance can become Active:

active
master
standby

There are only two scenarios that are analyzed by means of the traces:
- The startup trace is captured for both the Active and the Standby instance, and
- The manual graceful switchover from the Active to the Standby (endmqm -is QmgrName)

This document does not cover the trace of "failure" scenarios, such as network problems, disk problems, hardware outages, etc.

The queue manager in question is the only one active in both hosts for this scenario, in that way, there is no "contamination" in the traces from other queue managers.
The file system is NFS Version 4.

++ Acknowledgements

Thanks to Umamahesh Ponnuswamy for his assistance.

++ Related articles

https://www.ibm.com/support/pages/node/7112337
Using lslocks or lsof to list locked files when using IBM MQ multi-instance queue managers in Linux

https://www.ibm.com/support/pages/node/6348636
MQ Distributed: collection of articles regarding multi-instance queue managers

+ Zip file with trace files and miscellaneous

There is a companion zip file for this techdoc:
File-locking-multi-instance-93.zip

It contains the following files:
AMQ30939.0.EC.amqzxma0.host1.FMT
AMQ31110.0.dspmq.1.host1.FMT
AMQ31123.0.dspmq.2.host1.FMT
AMQ118435.0.EC.amqzxma0.host2.FMT
AMQ118462.0.dspmq.1.host2.FMT
AMQ118608.0.dspmq.2.host2.FMT
AMQERR01.locking.LOG
qmstatus.ini

++ Overview of the file locking mechanism to coordinate multiple instances

+ Configuration:

The hosts for this tutorial are:

Host1 (riggioni1):
The Active instance is started first.

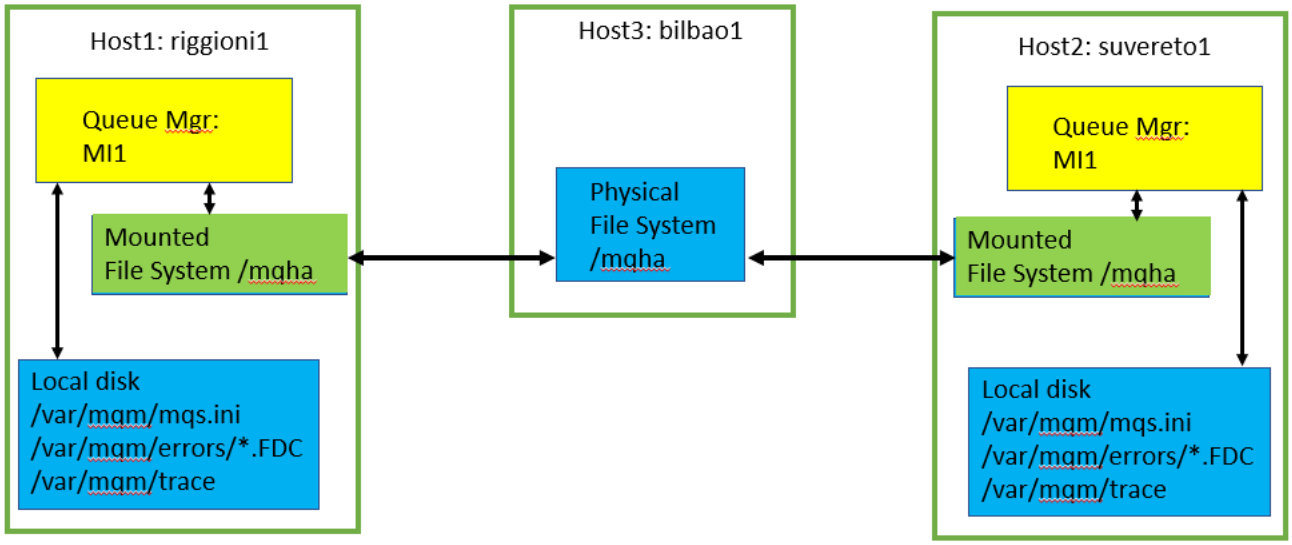
Host2 (suvereto1):
Then the Standby instance is started. It will become the active one after the switchover.

Host3 (bilbao1):
It is not involved in this scenario, but it is mentioned here for completeness. It has the physical file system for the queue manager data and recovery logs.

The hosts are described in this other tutorial:

https://www.ibm.com/support/pages/node/6985543
Configuring and using an IBM MQ 9.3 Multi-Instance in Linux

The following is an overview of the configuration:



Notice that host1 and host2 will have their own local /var/mqm directory structure which will be used for local queue managers, for the master index of the queue managers (mqs.ini), for the FDC files and for the trace files (*.TRC).

The mounted file system /mqha will contain the recovery/transaction logs for the queue managers, as well as the data for the queue managers, including the qm.ini configuration file and the error logs of the queue manager.

For this tutorial 3 Red Hat Enterprise Linux (RHEL) x86-64bit servers will be used.
The version is 8.6

The coordination mechanism of Active / Standby behavior is done via file locks on 3 shared files ("active", "master" and "standby") in the queue manager data directory:

```
mqm@riggioni1.fyre.ibm.com: /mqha/qmgrs/QMMI1
$ ls -l active master standby
-rw-rw-r-- 1 mqm mqm 4096 Aug 14 05:06 active
-rw-rw-r-- 1 mqm mqm 4096 Aug 14 05:06 master
-rw-rw-r-- 1 mqm mqm 4096 Aug 14 05:17 standby
```

The MQ strmqm command does not have a flag to indicate explicitly if a given instance is going to be labeled the "active" one.

Instead, there is a racing situation in which the "strmqm -x QMgr" which finishes first is the one that is going to be "active", and the other instance is going to be "standby".
Thus, if you want the active instance to be in host1, then you must issue "strmqm -x QMgr" in host1 and wait until it is fully started before you start the other instance in host2.

In this scenario, the first instance of a Multi-Instance queue manager is started via
strmqm -x QMGR1
such as in host1, then this instance is going to be the "active" one.
The other instance, the standby, is started in the same manner in host2.

The active instance of the queue manager requests the operating system to get a lock on the file named "master".
The operating system, on behalf of the queue manager, contacts the underlying file system to actually get a lock on the file.

<u>+ Notice that MQ is “file system agnostic”:</u>

- The queue manager does NOT have any knowledge on the nature of the actual file system being used: the queue manager contacts the operating system, the operating system does its magic and then returns information to the queue manager.

- The queue manager does NOT interact with the file system to get a file lock: instead, the queue manager contacts the operating system, and it is the responsibility of the operating system to interact with the file system. Then the file system returns the file lock (or the return code or error) to the operating system, and the operating system in turn communicates the result to the queue manager.

Note: This tutorial uses UNIX-style directory names, but the idea applies to Windows too.
For example, if in the /var/mqm/mqs.ini file the DataPath for the queue manager is:
  /mqha/qmgrs/QMMI1
Then the fully qualified name for this "master" file is:
  /mqha/qmgrs/QMMI1/master

The assumption is that /shared/qmdata is under NFS V4 control (UNIX) or a supported file system for Windows.
Again, the MQ queue manager does NOT know (and it does not care) that NFS or another supported file system is being used as a file system.

There are 3 files for the queue manager that are used for multi-instance:
  /mqha/qmgrs/QMMI1/master
  /mqha/qmgrs/QMMI1/active
  /mqha/qmgrs/QMMI1/standby

These 3 files contain text data in a single line (without a carriage return/line feed). The line has several tokens, the main ones are:
  item 1: hostname of the process that has the lock
  item 2: a lock id

The active instance also gets a lock on the file named "active", which provides another way to identify which is the active instance.

At this time, the active instance is running in host1 and the following files are locked:
  /mqha/qmgrs/QMMI1/master => locked by the active instance in host1
  /mqha/qmgrs/QMMI1/active => locked by the active instance in host1
  /mqha/qmgrs/QMMI1/standby => not locked

When the "standby" instance in host2 is started (strmqm -x QMGR1), then this queue manager will check if the lock for the "master" file has been taken already. In this scenario, it is "yes".

Because the standby instance cannot get a lock on the ‘master’ file, then the instance will get the lock for the corresponding file "standby", located in the same directory. The standby instance is not a functional queue manager.

At this time, the active instance is running in host1 and the standby instance is running in host2 and the following files are locked:
  /mqha/qmgrs/QMMI1/master => locked by the active instance in host1
  /mqha/qmgrs/QMMI1/active => locked by the active instance in host1
  /mqha/qmgrs/QMMI1/standby => locked by the standby instance in host2

The standby instance will try to acquire the lock for the file "master" every 2 seconds, and the file system will indicate to the standby instance that the file is locked by someone else.

If the lock for the file "master" is released, then when the standby instance asks again for the lock of the file "master", the file system will give the lock to the standby instance. Then, the standby instance will take the role of being the "active" instance and proceed to be a fully functional queue manager, capable to interact with clients.

It is important to notice that the active instance, having the lock for the file "master" is still going to ask periodically (every 10 seconds) if it still has the lock on the file "master". See item "c" below for more details on why this action is needed.

Based on the above, the most likely scenarios for an active instance to failover to a standby instance are discussed below.

The specific example of using NFS is illustrated, but the principle applies to other supported file systems.
(Again, the actual MQ code does NOT know which is the type of file system being used!!!)

a) The active instance is softly terminated by issuing "endmqm -is", indicating a "switchover", and the active instance will purposely release the lock for the file "master" (and the "active" file) and the standby will acquire the lock for both files to become the active instance and relinquish the lock for the file "standby" (allowing for another standby instance)
Note: the -i flag is for immediate and although strictly speaking is not necessary for multi-instance, it is a good idea to use it, because it will allow the instance to terminate even if there are connected clients (which is most likely to be the case).

b) The active instance is harshly terminated ("ps -9" or a crash of host1) and the instance does NOT have the time to release the file lock.
This is where NFS V3 will fail to do its job because NFS V3 will NOT release the lock acquired by the active instance (thus, NFS V3 is NOT supported due to its severe technical limitations).
Because NFS V4 works with "leased locks", the NFS server will determine that the process that acquired the lock for the file "master" has died and did not release the locks, and thus, the NFS server will break the lease of the lock with respect to the active instance.
Then, the standby instance interrogates the file system again if the lock is available for the file "master", the file system will reply that yes, it is available and the standby instance will grab the lock and become the active instance.

c) The network connection between the host1 (active instance) and the NFS server fails. In this case, the active instance when it checks again with the operating system to see if it has the lock on the file "master", the operating system (interacting with the NFS client - NFS server) will indicate to the active instance that it cannot get the status (no network connection). Thus, the active instance jumps to the conclusion that there is a problem and terminates.

Assuming that the network connection is still healthy between the host2 (standby instance) and the NFS server, then the NFS server will determine that the active instance cannot use the file "master" and thus will release the leased lock for the file "master". This will allow the standby instance to grab the lock on this file.

The successful operation of the MQ multi-instance feature critically depends on the correct functioning of NFS file locks. It also requires that operations (such as open and read) on an NFS file-locked file does not take longer than tens of seconds. If the queue manager does incur delays of tens of seconds then the queue manager has to conclude that the system is not operating correctly and the queue manager will shutdown to allow the standby instance to take over.

d) The queue manager has a couple of process threads that monitor the responsiveness of the operating system while performing the tasks asked by the queue manager (getting a file lock, inquiring the status of a lock, etc). If these threads determine that the response is not suitable, then the code will generate an FDC with more details, such as Probeid ZX155001 component zxcFileLockMonitorThread error lrcE_S_Q_MGR_UNRESPONSIVE.

For example, the FDC files reveal that due to an infrastructure problem (outside the scope of MQ), the active instance of the MQ queue manager could not get response from the underlying resources (most likely due to the file system), and thus, the active instance terminated, allowing the standby instance to become Active. The MQ code has worked as designed: to do a failover in case that there are problems with the file system, operating system or network system.

The following technote has more explanation about the unresponsiveness behavior from the underlying file system and/or disk, and why the MQ queue manager cannot do anything about it.

https://www.ibm.com/support/pages/node/473409
AMQ7280 queue manager appears unresponsive, Probeid ZX155001 component zxcFileLockMonitorThread error lrcE_S_Q_MGR_UNRESPONSIVE

+ Overall scenario:

Step 1: host1 - Start the Active: strmqm -x QMMI1
  trace file: AMQ30939.0.EC.amqzxma0.host1.FMT

Step 2: host1 - dspmq -x -m QMMI1 (shows only Active in host1)
  trace file: AMQ31110.0.dspmq.1.host1.FMT => it is the first dspmq from host1

Step 3: host2 - Start the Standby: strmqm -x QMMI1
  trace file: AMQ118435.0.EC.amqzxma0.host2.FMT

Step 4: host2 - dspmq -x -m QMMI1 (shows both Active in host1 and Standby in host2)
  trace file: AMQ118462.0.dspmq.1.host2.FMT => it is the first dspmq from host2

Step 5: host1 - End the Active (Standby becomes new Active): endmqm -is QMMI1
  trace file: AMQ30939.0.EC.amqzxma0.host1.FMT
  trace file: AMQ118435.0.EC.amqzxma0.host2.FMT

Step 6: host1 - dspmq -x -m QMMI1 (shows only new Active now in host2)
  trace file: AMQ31123.0.dspmq.2.host1.FMT => it is the second dspmq from host1

Step 7: host2 - dspmq -x -m QMMI1 (shows only new Active now in host2)
  trace file: AMQ118608.0.dspmq.2.host2.FMT => it is the second dspmq from host2

+ Preliminary steps

- Login as an MQ administrator in each host (host1 and host2)

- Delete previous traces in both hosts.
Keep in mind that the trace files will NOT be located in the shared file system; instead they will be in the /var/mqm/trace directory local to each host.
  host1: rm /var/mqm/trace/*
  host2: rm /var/mqm/trace/*

-- Note: We need to obtain the "early trace" in order to capture the startup of the queue managers.

-- Stop the queue managers. Assuming that Active is in host1 and Standby in host2
  host1 (Active): endmqm -i QMMI1
  host2 (Standby): endmqm -x QMMI1

-- Start the “early” trace
  host1: strmqtrc -e -t all -t detail
  host2: strmqtrc -e -t all -t detail

+ Scenario

Step 1: Start the Active queue manager
  host1: strmqm -x QMMI1
mqm@host1: /var/mqm/trace
$ strmqm -x QMMI1
IBM MQ queue manager 'QMMI1' starting.
The queue manager is associated with installation 'Installation1'.
6 log records accessed on queue manager 'QMMI1' during the log replay phase.
Log replay for queue manager 'QMMI1' complete.
Transaction manager state recovered for queue manager 'QMMI1'.
Plain text communication is enabled.
IBM MQ queue manager 'QMMI1' started using V9.3.2.0.

Step 2: Display status of Active, before Standby is started
  host1: dspmq -xf -m QMMI1
QMNAME(QMMI1)   STATUS(Running)
    INSTANCE(riggioni1.fyre.ibm.com) MODE(Active)
        master(riggioni1.fyre.ibm.com,7225327309153730329)
        active(riggioni1.fyre.ibm.com,7225327309153730329)

Step 3: Start Standby
  host2: strmqm -x QMMI1
IBM MQ queue manager 'QMMI1' starting.
The queue manager is associated with installation 'Installation2'.
Plain text communication is enabled.
A standby instance of queue manager 'QMMI1' has been started. The active
instance is running elsewhere.

Step 4: Display the status of the queue managers before switchover
.
host2: dspmq -xf -m QMMI1

QMNAME(QMMI1)   STATUS(Running as standby)
    INSTANCE(riggioni1.fyre.ibm.com) MODE(Active)
    INSTANCE(suvereto1.fyre.ibm.com) MODE(Standby)
        master(riggioni1.fyre.ibm.com,7225327309153730329)
        active(riggioni1.fyre.ibm.com,7225327309153730329)
        standby(suvereto1.fyre.ibm.com,7225327562557880603)

Step 5: Do a controlled switchover:
  host1: endmqm -is QMMI1
IBM MQ queue manager 'QMMI1' ending.
IBM MQ queue manager 'QMMI1' ended, permitting switchover to a standby
instance.
Now the Standby instance in host2 should become the Active instance.

Step 6: From host1, display the status of the queue managers after the switchover has completed

```
 host1:  dspmq -xf -m QMMI1
QMNAME(QMMI1)                                 STATUS(Running elsewhere)
   INSTANCE(suvereto1.fyre.ibm.com) MODE(Active)
      master(suvereto1.fyre.ibm.com,7225327562557880603)
      active(suvereto1.fyre.ibm.com,7225327562557880603)
```

Step 7: From host2, display the status of the queue managers after the switchover has completed

```
 host2:  dspmq -xf -m QMMI1
QMNAME(QMMI1)                                 STATUS(Running)
   INSTANCE(suvereto1.fyre.ibm.com) MODE(Active)
      master(suvereto1.fyre.ibm.com,7225327562557880603)
      active(suvereto1.fyre.ibm.com,7225327562557880603)
```

+ Post-scenario steps

- End the traces

```
 host1:  endmqtrc -a
IBM MQ trace for installation 'Installation1' has stopped.

 host2:  endmqtrc -a
IBM MQ trace for installation 'Installation2' has stopped.
```

- Format the trace (resulting in human readable files that have a suffix of FMT)
  This step is applicable only for Unix. The files in Windows are formatted already.

```
 host1:
      cd /var/mqm/trace/
      dspmqtrc *.TRC

 host2:
      cd /var/mqm/trace/
      dspmqtrc *.TRC
```

- Prepare tar files, compress them and send them via ftp. The following command will gather also the FDC file and the error logs.

The zip file will be generated in the current directory (flag: outputdir).

From host1:
```
cd /home/mqm
runmqras -qmlist QMMI1 -section trace -outputdir .
```

From host2:
```
cd /home/mqm
runmqras -qmlist QMMI1 -section trace -outputdir .
```

Note:
The output of runmqras includes the file with the output of "mount.stdout".
We can see that they entry for /mqha is type nfs4
mount.stdout
9.46.80.xx:/ on /mqha **type nfs4**
(rw,relatime,vers=4.2,rsize=262144,wsize=262144,namlen=255,hard,proto=tcp,timeo=600,retrans=2,sec=sys,clientaddr=9.46.66.142,local_lock=none,addr=9.46.80.75)

++ Reviewing traces

+ Step 1: Starting Active instance

Status at start of this step:
host1: No instance running
host2: No instance running
master file locked by: none
active file locked by: none
standby file locked by: none

It is necessary to identify the trace file for the Execution Controller (amqzxma0).
Why? because it is the one that handles the file locking for the files: master, active, standby
Note: The trace files for dspmq will also be needed later on, because it queries the status of the locks.

The trace files have a header with identifying information and one of the lines is "Program Name", such as:
| Program Name :- amqzxma0

Thus, let's find out the trace files for amqzxma0 and dspmq:

```
$ cd /var/mqm/trace
$ grep "Program Name" *.FMT | grep amqzxma0
AMQ30939.0.FMT:| Program Name      :- amqzxma0
```

Note: Because it is difficult to remember PIDs, thus, the file is going to be manually renamed to make it easier to identify it:
AMQ30939.0.EC.amqzxma0.host1.FMT => it is the Execution Controller for host1 (Step 1)

```
$ grep "Program Name" *.FMT | grep dspmq
AMQ31110.0.FMT:| Program Name      :- dspmq
AMQ31123.0.FMT:| Program Name      :- dspmq
```

Note: Similarly the files were renamed:
AMQ31110.0.dspmq.1.host1.FMT => it is the first dspmq from host1 (Step 2)
AMQ31123.0.dspmq.2.host1.FMT => it is the second dspmq from host1 (Step 6)

+ Note about the values for LockMode found in the trace:

The source code header file amqxnfla.h has the following regarding LockMode:
```
 #define xcsFILE_LOCK_NO_LOCK    0x0000
 #define xcsFILE_LOCK_EXCLUSIVE  0x0001
 #define xcsFILE_LOCK_SHARED     0x0002
```

+ host1: Reviewing file: file AMQ30939.0.EC.amqzxma0.host1.FMT

It is faster to reach to the section that checks for the file locks, by searching for the function:
```
  zutRequestQMFileLocks
```

This is the start of the block that handles the file locks
```
 12:20:08.155561    30939.1         :      --{  zutRequestQMFileLocks
 12:20:08.155565    30939.1         :          DataPath(/mqha/qmgrs/QMMI1) Flags(0X41)
 12:20:08.155567    30939.1         :      ---{  xcsIsEnvironment
 12:20:08.155571    30939.1         :          xcsIsEnvironment[AMQ_FILE_LOCKS_OFF] =
FALSE
 12:20:08.155573    30939.1         :      ---}  xcsIsEnvironment rc=OK FunctionTime=6
```

The queue manager tries to get a file lock xcsFILE_LOCK_EXCLUSIVE on the file "master".
```
 12:20:08.155608    30939.1         :      ---{  xcsOpenFileLock
 12:20:08.155611    30939.1         :          FileName(/mqha/qmgrs/QMMI1/master)
LockMode(0X1)
 12:20:08.164879    30939.1         :      ----{  xcsWaitFileLock
 12:20:08.164895    30939.1         :          fd(13) LockType(0X1) timeout(0)
 12:20:08.165216    30939.1         :      ----}  xcsWaitFileLock rc=OK FunctionTime=337
```

If the lock is acquired, then write into that file.
```
 12:20:08.165230    30939.1         :          Lock details to write into master:
riggioni1.fyre.ibm.com,7225327309153730329,1682277608,501,501
 12:20:08.165235    30939.1         :      ---{  xcsWriteFileLock
 12:20:08.165237    30939.1         :
LockData(riggioni1.fyre.ibm.com,7225327309153730329,1682277608,501,501)
 12:20:08.178946    30939.1         :      ---}  xcsWriteFileLock rc=OK FunctionTime=13711
```

The queue manager tries to get a file lock xcsFILE_LOCK_EXCLUSIVE on the file "active".
```
 12:20:08.178973    30939.1         :      ---{  xcsOpenFileLock
 12:20:08.178978    30939.1         :          FileName(/mqha/qmgrs/QMMI1/active)
LockMode(0X1)
 12:20:08.180541    30939.1         :      ----{  xcsWaitFileLock
 12:20:08.180551    30939.1         :          fd(14) LockType(0X1) timeout(0)
 12:20:08.180947    30939.1         :      ----}  xcsWaitFileLock rc=OK FunctionTime=406
 12:20:08.180953    30939.1         :          fd(14)
 12:20:08.180955    30939.1         :      ---}  xcsOpenFileLock rc=OK FunctionTime=1982
```

If the lock is acquired, then write into that file.

```
 12:20:08.180958    30939.1         :          Lock details to write into active:
riggioni1.fyre.ibm.com,7225327309153730329,1682277608,501,501
 12:20:08.180960    30939.1         :      ---{  xcsWriteFileLock
 12:20:08.180962    30939.1         :
LockData(riggioni1.fyre.ibm.com,7225327309153730329,1682277608,501,501)
 12:20:08.192574    30939.1         :      ---}  xcsWriteFileLock rc=OK FunctionTime=11614
```

Closing the file lock for the active:

```
 12:20:08.192596    30939.1         :      ---{  xcsCloseFileLock
 12:20:08.193189    30939.1         :          Data: 0x0000000e 0x00000000
 12:20:08.193706    30939.1         :      ---}  xcsCloseFileLock rc=OK FunctionTime=1110
```

The queue manager tries to get a file lock on the file "active". This time the lock mode is different: xcsFILE_LOCK_SHARED

```
 12:20:08.193715    30939.1         :      ---{  xcsOpenFileLock
 12:20:08.193720    30939.1         :          FileName(/mqha/qmgrs/QMMI1/active)
LockMode(0X2)
 12:20:08.194437    30939.1         :      ----{  xcsWaitFileLock
 12:20:08.194443    30939.1         :          fd(14) LockType(0X2) timeout(0)
 12:20:08.194779    30939.1         :      ----}  xcsWaitFileLock rc=OK FunctionTime=342
 12:20:08.194789    30939.1         :          fd(14)
 12:20:08.194791    30939.1         :      ---}  xcsOpenFileLock rc=OK FunctionTime=1076
 12:20:08.194795    30939.1         :      ---{  xcsReadFileLock
 12:20:08.195235    30939.1         :      ---}  xcsReadFileLock rc=OK FunctionTime=440
 12:20:08.195244    30939.1         :          Lock details read from active:
riggioni1.fyre.ibm.com,7225327309153730329,1682277608,501,501
 12:20:08.195263    30939.1         :      --}  zutRequestQMFileLocks rc=OK
FunctionTime=39702
```

Checks for the lock status for "standby" to ensure that it is available (it does not get a lock and it does not try to write into that file).

```
 12:20:08.195268    30939.1         :      --{  zutRequestQMFileLocks
 12:20:08.195272    30939.1         :          DataPath(/mqha/qmgrs/QMMI1) Flags(0X1200)
 12:20:08.195276    30939.1         :      ---{  xcsOpenFileLock
 12:20:08.195278    30939.1         :          FileName(/mqha/qmgrs/QMMI1/standby)
LockMode(0)
 12:20:08.195846    30939.1         :          fd(15)
 12:20:08.195854    30939.1         :      ---}  xcsOpenFileLock rc=OK FunctionTime=578
 12:20:08.195857    30939.1         :      ---{  xcsCloseFileLock
 12:20:08.195865    30939.1         :          Data: 0x0000000f 0x00000000
 12:20:08.196098    30939.1         :      ---}  xcsCloseFileLock rc=OK FunctionTime=241
 12:20:08.196102    30939.1         :      --}  zutRequestQMFileLocks rc=OK
FunctionTime=834
```

At this point, the status (LockMode) for each file is:
  FileName(/mqha/qmgrs/QMMI1/master) LockMode(0X1) => exclusive (by host1 riggioni1)
  FileName(/mqha/qmgrs/QMMI1/active) LockMode(0X2) => shared (by host1 riggioni1)
  FileName(/mqha/qmgrs/QMMI1/standby) LockMode(0) => not locked

Even though it is not related to file locking, because the Active was started with "strmqm -x", then the qmstatus.ini is updated to include:
  PermitStandby=Yes

```
 12:20:08.239122   30939.1        :     -----{  xusWriteSingleStanza
 12:20:08.239125   30939.1        :         QueueManagerStatus:
 12:20:08.239128   30939.1        :           CurrentStatus=Starting
 12:20:08.239130   30939.1        :           PermitStandby=Yes
 12:20:08.239133   30939.1        :           PermitFailover=Yes
 12:20:08.239135   30939.1        :           PlatformSignature=20497
 12:20:08.239137   30939.1        :           PlatformString=Linux 4.18.0-
425.19.2.el8_7.x86_64
 12:20:08.239140   30939.1        :           UpdateTime=2023-04-23 12:20:08.238955
 12:20:08.239142   30939.1        :           Version=9.3.2.0
 12:20:08.239144   30939.1        :           MinimumRequiredVersion=9.3.2.0
 12:20:08.239147   30939.1        :         RetCode (OK)
 12:20:08.239148   30939.1        :     -----}  xusWriteSingleStanza rc=OK FunctionTime=26
```

+ After the Active instance has started notice the following in the "master" and "active" files.

This is the contents of the master and active files:

host1
$ cat /mqha/qmgrs/QMMI1/master
cat /mqha/qmgrs/QMMI1/master
riggioni1.fyre.ibm.com,7225330354288567839,1682278317,501,501

$ cat /mqha/qmgrs/QMMI1/active
riggioni1.fyre.ibm.com,7225330354288567839,1682278317,501,501

The standby file will be discussed later on.

The contents of each file is 1 line.
This is a reference to the tokens that compose the line:

The master, active and standby files contain a little data about the lock holder:
- Hostname
- Lock id – identifies the queue manager instance
  When an instance starts, it calculates the lock id which it writes into the lock files that it owns
- Lock time
- User id of MQ administrator "mqm" in Unix
- Group id of MQ administrator "mqm" in Unix

- Notice a change to qmstatus.ini

Because the flag "-x" was used with strmqm, the qmstatus.ini is updated to indicate that a Standby instance is allowed:
QueueManagerStatus:
  PermitStandby=Yes

mqm@host1:
$ head /mqha/qmgrs/QMMI1/qmstatus.ini
AuthorityData:
  Creator=mqm
QueueManagerStatus:
  CurrentStatus=Running
  PermitStandby=Yes
  PermitFailover=Yes
  PlatformSignature=20497
  PlatformString=Linux 4.18.0-425.19.2.el8_7.x86_64
  Version=9.3.2.0
  MinimumRequiredVersion=9.3.2.0

+ Message in error log of the queue manager indicating that the Active queue manager has started:

Notice that the Host(riggioni1) is mentioned in the header.
There is no explicit indication that it is the "Active", but implicitly, it is the Active one.

directory: /mqha/qmgrs/QMMI1/errors
File: AMQERR01.LOG

04/23/2023 12:20:09 PM - Process(30939.1) User(mqm) Program(amqzxma0)
Host(riggioni1.fyre.ibm.com) Installation(Installation1)
VRMF(9.3.2.0) QMgr(QMMI1)
Time(2023-04-23T19:20:09.933Z)
CommentInsert1(9.3.2.0)
CommentInsert3(QMMI1)
AMQ8003I: IBM MQ queue manager 'QMMI1' started using V9.3.2.0.
EXPLANATION:
IBM MQ queue manager 'QMMI1' started using V9.3.2.0.

+ Step 2: host1 - dspmq (shows only Active)

+ Question: How do we know that this host is the Active?

Issue the following dspmq command.

```
mqm@host1: /var/mqm/trace
$ dspmq -xf -m QMMI1
QMNAME(QMMI1)                                             STATUS(Running)
    INSTANCE(riggioni1.fyre.ibm.com) MODE(Active)
        master(riggioni1.fyre.ibm.com,7225327309153730329)
        active(riggioni1.fyre.ibm.com,7225327309153730329)
```

Notice: MODE(Active)

Let's review the trace, searching for:
zutQueryQMFileLocks
File: AMQ31110.0.dspmq.1.host1.FMT

```
 12:20:34.758919    31110.1          :      -{  zutQueryQMFileLocks
 12:20:34.758922    31110.1          :          DataPath(/mqha/qmgrs/QMMI1) Flags(0X1)
 12:20:34.758925    31110.1          :      --{  xcsIsEnvironment
 12:20:34.758927    31110.1          :          xcsIsEnvironment[AMQ_FILE_LOCKS_OFF] =
FALSE
 12:20:34.758929    31110.1          :      --}  xcsIsEnvironment rc=OK FunctionTime=4
```

The file "master" is locked:

```
 12:20:34.758933    31110.1          :      --{  xcsQueryFileLock
 12:20:34.758936    31110.1          :          FileName(/mqha/qmgrs/QMMI1/master)
 12:20:34.760231    31110.1          :          Lock data read ()
 12:20:34.760246    31110.1          :          Data: 0x00000008 0x00000000
 12:20:34.760260    31110.1          :      --}  xcsQueryFileLock rc=OK FunctionTime=1327
 12:20:34.760273    31110.1          :          Lock data for /mqha/qmgrs/QMMI1/master
is <riggioni1.fyre.ibm.com> <LOCKED>
 12:20:34.760275    31110.1          :      -}  zutQueryQMFileLocks rc=OK FunctionTime=1356
```

The file "active" is locked:

```
12:20:34.760278    31110.1          :      -{  zutQueryQMFileLocks
 12:20:34.760280    31110.1          :          DataPath(/mqha/qmgrs/QMMI1) Flags(0X2)
 12:20:34.760283    31110.1          :      --{  xcsQueryFileLock
 12:20:34.760285    31110.1          :          FileName(/mqha/qmgrs/QMMI1/active)
 12:20:34.761199    31110.1          :          Lock data read ()
 12:20:34.761211    31110.1          :          Data: 0x00000008 0x00000000
 12:20:34.761712    31110.1          :      --}  xcsQueryFileLock rc=OK FunctionTime=1429
 12:20:34.761725    31110.1          :          Lock data for /mqha/qmgrs/QMMI1/active is
<riggioni1.fyre.ibm.com> <LOCKED>
```

```
 12:20:34.761727    31110.1         :      -}  zutQueryQMFileLocks rc=OK FunctionTime=1449
```

The file "standby" is NOT locked:

```
 12:20:34.761730    31110.1         :      -{  zutQueryQMFileLocks
 12:20:34.761732    31110.1         :          DataPath(/mqha/qmgrs/QMMI1) Flags(0X3)
 12:20:34.761735    31110.1         :      --{  xcsQueryFileLock
 12:20:34.761738    31110.1         :          FileName(/mqha/qmgrs/QMMI1/standby)
 12:20:34.762833    31110.1         :          Lock data read ()
 12:20:34.762845    31110.1         :          Data: 0x00000008 0x00000000
 12:20:34.763126    31110.1         :      --}  xcsQueryFileLock rc=OK FunctionTime=1391
 12:20:34.763136    31110.1         :          Lock data for /mqha/qmgrs/QMMI1/standby
is <riggioni1.fyre.ibm.com> <NOT LOCKED>
 12:20:34.763138    31110.1         :      -}  zutQueryQMFileLocks rc=OK FunctionTime=1408
```

+ Question: Is there a way to find out if the "active" file is locked without taking the MQ trace?

The full path of the file is:
/mqha/qmgrs/QMMI1/active

The Unix tool "lsof" can be used to show the file locks.

lsof is a tool that shows only the LOCAL processes that have locked a file.
In other words, if you login to the host that has the NFS server, lsof will NOT show those files that have been locked in remote hosts where the NFS file system has been mounted.
You will need to visit each host to find out the locked files.

Login as root in order to execute the operating system "lsof" to find out which applications have a lock on the files
(if you can execute successfully "lsof" as user "mqm" or an MQ administrator, then it is OK to proceed with that userid).

In this scenario, the Active instance is running in host1

The meaning of the 4th column (FD) is the one that is of interest for us regarding file locking:
If the first letter is the lower case "r" means: r for read access;
If the first letter is the lower case "u" means: u for read and write access;
If the last character is the upper case "W" means: W for a write lock on the entire file;
If the last character is the upper case"R" means: R for a read lock on the entire file;
If the last character is a space, then there is no lock.

Thus:
uW means - read and write access, write lock on entire file
rW means - read access, write lock on entire file
rR means - read access, read lock on entire file
r means - read access, no lock

In host1 where the Active instance is running, issue the following to list the locks for the 'master' file.
Under the column FD notice the value of uW:
uW => which means that the file is opened with read and write access, and with write lock on entire file

```
ROOT@host1-riggioni1.fyre.ibm.com: /root
# lsof /mqha/qmgrs/QMMI1/master
COMMAND    PID USER   FD   TYPE DEVICE SIZE/OFF      NODE NAME
amqzxma0 30939  mqm   18uW  REG   0,48     4096 134548712 /mqha/qmgrs/QMMI1/master
```

List the locks for the 'active' file.
Under the column FD notice the value of rR:
rR => which means that the file is opened with read access mode, and with read lock on entire file

```
ROOT@host1-riggioni1.fyre.ibm.com: /root
# lsof /mqha/qmgrs/QMMI1/active
COMMAND    PID USER   FD   TYPE DEVICE SIZE/OFF      NODE NAME
amqzxma0 30939  mqm   19rR  REG   0,48     4096 134548713 /mqha/qmgrs/QMMI1/active
amqzfuma 35020  mqm    9rR  REG   0,48     4096 134548713 /mqha/qmgrs/QMMI1/active
amqzmuc0 35027  mqm   10rR  REG   0,48     4096 134548713 /mqha/qmgrs/QMMI1/active
amqzmuc0 35027  mqm   11rR  REG   0,48     4096 134548713 /mqha/qmgrs/QMMI1/active
amqzmuc0 35027  mqm   16rR  REG   0,48     4096 134548713 /mqha/qmgrs/QMMI1/active
amqzmuf0 35073  mqm    6rR  REG   0,48     4096 134548713 /mqha/qmgrs/QMMI1/active
amqrrmfa 35077  mqm    6rR  REG   0,48     4096 134548713 /mqha/qmgrs/QMMI1/active
amqzlaa0 35088  mqm    6rR  REG   0,48     4096 134548713 /mqha/qmgrs/QMMI1/active
amqfqpub 35132  mqm    6rR  REG   0,48     4096 134548713 /mqha/qmgrs/QMMI1/active
java     35148  mqm   93rR  REG   0,48     4096 134548713 /mqha/qmgrs/QMMI1/active
amqfcxba 35162  mqm    6rR  REG   0,48     4096 134548713 /mqha/qmgrs/QMMI1/active
```

Notice that the Execution Controller (EC) for the queue manager is called "amqzxma0":
amqzxma0 30939 mqm 19rR REG 0,48 4096 134548713 /mqha/qmgrs/QMMI1/active

You can use the following to find out the command and arguments for the EC:

```
mqm@host1: /mqha/qmgrs/QMMI1/
$ ps -ef | grep 35010
mqm      30939 1  0 04:00 ?        00:00:00 /opt/mqm/bin/amqzxma0 -m QMMI1 -x -u mqm
```

Show the contents of the "active" file. Notice the 1st argument, which is the host name.
cat /mqha/qmgrs/QMMI1/**active**
riggioni1.fyre.ibm.com,7225330354288567839,1682278317,501,501

List the locks for the 'standby' file.
There are no locks on this file from host1.

lsof /mqha/qmgrs/QMMI1/standby
(none)

+ Step 3: Start the Standby instance:

Status at start of this step:
host1: Active
host2: No instance running
master file locked by: host1
active file locked by: host1
standby file locked by: none

We need to review now the traces from host2

In the same way that we identified the trace files for the EC and the dspmq for the Active in host1, the corresponding files for the Standby in host2 were identified and renamed:

AMQ118435.0.EC.amqzxma0.host2.FMT => it is the Execution Controller for host2 (Step 3)
AMQ118462.0.dspmq.1.host2.FMT => it is the first dspmq from host2 (Step 4)
AMQ118608.0.dspmq.2.host2.FMT => it is the second dspmq from host2 (Step 7)

Reviewing trace for EC:
File: AMQ118435.0.EC.amqzxma0.host2.FMT

Search for: zutRequestQMFileLocks

```
 12:21:07.419064   118435.1         :      --{  zutRequestQMFileLocks
 12:21:07.419076   118435.1         :          DataPath(/mqha/qmgrs/QMMI1) Flags(0X41)
```

Check if master file is available by trying to get a lock

```
 12:21:07.419205   118435.1         :      ---{  xcsOpenFileLock
 12:21:07.419212   118435.1         :          FileName(/mqha/qmgrs/QMMI1/master)
LockMode(0X1)
 12:21:07.421631   118435.1         :      ----{  xcsWaitFileLock
 12:21:07.421690   118435.1         :          fd(13) LockType(0X1) timeout(0)
 12:21:07.423225   118435.1         :          Data: 0x0000000b 0x00000001 0x00000000
0x00000001
 12:21:07.423265   118435.1         :          Lock not granted
 12:21:07.423275   118435.1         :      ----}! xcsWaitFileLock
rc=xecN_E_LOCK_NOT_GRANTED FunctionTime=1644
 12:21:07.423284   118435.1         :          Data: 0x0000000d 0x00000000
 12:21:07.423868   118435.1         :          fd(-1)
 12:21:07.423898   118435.1         :      ---}! xcsOpenFileLock
rc=xecN_E_LOCK_NOT_GRANTED FunctionTime=4693
 12:21:07.423907   118435.1         :      --}! zutRequestQMFileLocks
rc=lpiRC_Q_MGR_LOCK_UNAVAILABLE FunctionTime=4843
```

Notice that the lock was not granted, thus, this means that the Active instance is already running and has locked the file!
Now the queue manager will try to grab the standby file

```
 12:21:07.423917   118435.1         :      --{  zutRequestQMFileLocks
 12:21:07.423925   118435.1         :          DataPath(/mqha/qmgrs/QMMI1) Flags(0X1200)
 12:21:07.423935   118435.1         :      ---{  xcsOpenFileLock
 12:21:07.423942   118435.1         :          FileName(/mqha/qmgrs/QMMI1/standby)
LockMode(0)
 12:21:07.424949   118435.1         :          fd(13)
 12:21:07.424974   118435.1         :      ---}  xcsOpenFileLock rc=OK FunctionTime=1039
 12:21:07.424991   118435.1         :      ---{  xcsCloseFileLock
 12:21:07.425017   118435.1         :          Data: 0x0000000d 0x00000000
 12:21:07.425494   118435.1         :      ---}  xcsCloseFileLock rc=OK FunctionTime=503
 12:21:07.425509   118435.1         :      --}  zutRequestQMFileLocks rc=OK
FunctionTime=1592
```

The queue manager was able to get the standby file.

```
 12:21:07.425574   118435.1         :      ---{  xcsOpenFileLock
 12:21:07.425579   118435.1         :          FileName(/mqha/qmgrs/QMMI1/standby)
LockMode(0X1)
 12:21:07.426491   118435.1         :      ----{  xcsWaitFileLock
 12:21:07.426512   118435.1         :          fd(13) LockType(0X1) timeout(0)
 12:21:07.427487   118435.1         :      ----}  xcsWaitFileLock rc=OK FunctionTime=996
 12:21:07.427514   118435.1         :          fd(13)
 12:21:07.427519   118435.1         :      ---}  xcsOpenFileLock rc=OK FunctionTime=1945
 12:21:07.427525   118435.1         :          Lock details to write into standby:
suvereto1.fyre.ibm.com,7225327562557880603,1682277667,501,501
 12:21:07.427535   118435.1         :      ---{  xcsWriteFileLock
 12:21:07.427540   118435.1         :
LockData(suvereto1.fyre.ibm.com,7225327562557880603,1682277667,501,501)
 12:21:07.442873   118435.1         :      ---}  xcsWriteFileLock rc=OK FunctionTime=15338
```

Even though it is not related to file locking, the Standby instance needs to verify if qmstatus.ini indicates that a Standby is permitted. If yes (Active used "strmqm -x"), then continue.
If not, then stop (that is, the Active did not specify the -x in strmqm).
   PermitStandby=Yes

```
12:21:55.458118   118435.1         :      -----{  xusWriteSingleStanza
 12:21:55.458124   118435.1         :          QueueManagerStatus:
 12:21:55.458129   118435.1         :            CurrentStatus=Starting
 12:21:55.458133   118435.1         :            PermitStandby=Yes
 12:21:55.458137   118435.1         :            PermitFailover=Yes
 12:21:55.458141   118435.1         :            PlatformSignature=20497
```

```
 12:21:55.458146   118435.1         :            PlatformString=Linux 4.18.0-
425.19.2.el8_7.x86_64
 12:21:55.458155   118435.1         :            UpdateTime=2023-04-23 12:21:55.457815
 12:21:55.458159   118435.1         :            Version=9.3.2.0
 12:21:55.458163   118435.1         :            MinimumRequiredVersion=9.3.2.0
 12:21:55.458167   118435.1         :          RetCode (OK)
```

The queue manager starts in Standby mode

```
mqm@host2: /var/mqm/trace
$ dspmq -xf -m QMMI1
QMNAME(QMMI1)                                             STATUS(Running as standby)
   INSTANCE(riggioni1.fyre.ibm.com) MODE(Active)
   INSTANCE(suvereto1.fyre.ibm.com) MODE(Standby)
      master(riggioni1.fyre.ibm.com,7225327309153730329)
      active(riggioni1.fyre.ibm.com,7225327309153730329)
      standby(suvereto1.fyre.ibm.com,7225327562557880603)
```

Notice that there are only 4 processes running:

```
mqm@host2: /var/mqm/trace
$ ps -ef | grep -i mq
mqm      23158      1  0 12:35 ?       00:00:00 /opt/mqm93/bin/amqzxma0 -m QMMI1 -x -u
mqm
mqm      122429  23158  0 12:35 ?       00:00:00 /opt/mqm93/bin/amqzfuma -m QMMI1
mqm      122433  23158  0 12:35 ?       00:00:00 /opt/mqm93/bin/amqzmgr0 -m QMMI1
mqm      122436  23158  0 12:35 ?       00:00:00 /opt/mqm93/bin/amqzmuc0 -m QMMI1
```

Let's look at the status of the locks:

```
ROOT@host2: /root
# lsof /mqha/qmgrs/QMMI1/master
COMMAND    PID USER   FD   TYPE DEVICE SIZE/OFF      NODE NAME
amqzxma0 23158  mqm   18u   REG   0,48     4096 134548712 /mqha/qmgrs/QMMI1/master

ROOT@host2: /root
# lsof /mqha/qmgrs/QMMI1/active
Note from author: no output
```

There will be a lock on the file "standby"
The value for FD includes rW and uW:
rW means - read access, write lock on entire file
uW means - read and write access, write lock on entire file

ROOT@host2: /root
lsof /mqha/qmgrs/QMMI1/standby
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
amqzxma0 23158 mqm 10uW REG 0,48 4096 134551303
/mqha/qmgrs/QMMI1/standby

The contents of the standby file is:

cat /mqha/qmgrs/QMMI1/**standby**
suvereto1.fyre.ibm.com,7225331161741564019,1682278505,501,501

+ Notice new entry in the error log:

directory: /mqha/qmgrs/QMMI1/errors
File: AMQERR01.LOG

Notice that this is an explicit indication that the Standby instance was started, and the Host name is mentioned in the header:

04/23/2023 12:21:07 PM - Process(118435.1) User(mqm) Program(amqzxma0)
Host(suvereto1.fyre.ibm.com) Installation(Installation2)
VRMF(9.3.2.0) QMgr(QMMI1)
Time(2023-04-23T19:21:07.495Z)
CommentInsert3(QMMI1)
AMQ8060I: IBM MQ queue manager 'QMMI1' started as a standby instance.
EXPLANATION:
Queue manager 'QMMI1' started as a standby instance, ready to become the active instance if the existing active instance fails.

+ Step 4: Steady state - dspmq from host2 and review of file locks

Status at start of this step:
host1: Active
host2: Standby
master file locked by: host1
   Lock data for /mqha/qmgrs/QMMI1/master is <riggioni1> <LOCKED>
active file locked by: host1
   Lock data for /mqha/qmgrs/QMMI1/active is <riggioni1> <LOCKED>
standby file locked by: host2
   Lock data for /mqha/qmgrs/QMMI1/standby is <suvereto1> <LOCKED>

Let's review the trace for the dspmq command that shows that the Active is in host1 and the Standby is in host2:

File: AMQ118462.0.dspmq.1.host2.FMT

Notice that the 3 files are locked:

```
 12:21:32.543584   118462.1         :     --{  xcsQueryFileLock
 12:21:32.543588   118462.1         :          FileName(/mqha/qmgrs/QMMI1/master)
 12:21:32.546478   118462.1         :          Lock data read ()
 12:21:32.546523   118462.1         :          Data: 0x00000008 0x00000000
 12:21:32.546541   118462.1         :     --}  xcsQueryFileLock rc=OK FunctionTime=2957
 12:21:32.546567   118462.1         :          Lock data for /mqha/qmgrs/QMMI1/master
is <riggioni1.fyre.ibm.com> <LOCKED>
 12:21:32.546572   118462.1         :     -}  zutQueryQMFileLocks rc=OK FunctionTime=3205

 12:21:32.546589   118462.1         :     --{  xcsQueryFileLock
 12:21:32.546594   118462.1         :          FileName(/mqha/qmgrs/QMMI1/active)
 12:21:32.549462   118462.1         :          Lock data read ()
 12:21:32.549527   118462.1         :          Data: 0x00000008 0x00000000
 12:21:32.553535   118462.1         :     --}  xcsQueryFileLock rc=OK FunctionTime=6946
 12:21:32.553596   118462.1         :          Lock data for /mqha/qmgrs/QMMI1/active is
<riggioni1.fyre.ibm.com> <LOCKED>
 12:21:32.553601   118462.1         :     -}  zutQueryQMFileLocks rc=OK FunctionTime=7023

12:21:32.553620   118462.1         :     --{  xcsQueryFileLock
 12:21:32.553625   118462.1         :          FileName(/mqha/qmgrs/QMMI1/standby)
 12:21:32.555011   118462.1         :          Lock data read ()
 12:21:32.555044   118462.1         :          Data: 0x00000008 0x00000000
 12:21:32.555072   118462.1         :     --}  xcsQueryFileLock rc=OK FunctionTime=1452
 12:21:32.555084   118462.1         :          Lock data for /mqha/qmgrs/QMMI1/standby
is <suvereto1.fyre.ibm.com> <LOCKED>
 12:21:32.555088   118462.1         :     -}  zutQueryQMFileLocks rc=OK FunctionTime=1480
```

+ Note about the monitoring threads

The names of the monitoring threads are:
zxcFileLockVerifyThread
zxcStartFileLockMonitorThread

At this time, these monitoring threads did not have an active role because the "endmqm -is" command was used, which did a graceful switchover.
But if there was a problem with the responsiveness/availability of the network/filesystem, then these monitoring threads may force a failover (generating FDCs).

host1:
In the trace for the Active instance, here is the information about the monitoring threads:

```
 12:20:10.579062    30939.15         :      -{  zxcFileLockVerifyThread
..
 12:20:10.569644    30939.1          :      ---{  zxcStartFileLockMonitorThread

 12:20:10.579125    30939.1          :      ---}  zxcStartFileLockMonitorThread rc=OK
FunctionTime=139
```

Host2:
Notice that the Standby EC is checking the file lock for "master" every 2 seconds:

```
 12:21:07.419205   118435.1          :      ---{  xcsOpenFileLock
 12:21:07.419212   118435.1          :          FileName(/mqha/qmgrs/QMMI1/master)
LockMode(0X1)
 12:21:07.421631   118435.1          :      ----{  xcsWaitFileLock
 12:21:07.421690   118435.1          :          fd(13) LockType(0X1) timeout(0)
 12:21:07.423225   118435.1          :          Data: 0x0000000b 0x00000001 0x00000000
0x00000001
 12:21:07.423265   118435.1          :          Lock not granted
 12:21:07.423275   118435.1          :      ----}! xcsWaitFileLock
rc=xecN_E_LOCK_NOT_GRANTED FunctionTime=1644
12:21:07.423898   118435.1          :      ---}! xcsOpenFileLock
rc=xecN_E_LOCK_NOT_GRANTED FunctionTime=4693
 12:21:07.423907   118435.1          :      --}! zutRequestQMFileLocks
rc=lpiRC_Q_MGR_LOCK_UNAVAILABLE FunctionTime=4843
…
 12:23:28.547288    118435.1          :      ---}! xcsOpenFileLock
rc=xecN_E_LOCK_NOT_GRANTED FunctionTime=1411
 12:23:28.547295    118435.1          :      --}! zutRequestQMFileLocks
rc=lpiRC_Q_MGR_LOCK_UNAVAILABLE FunctionTime=1444
```

+ Step 5-a: Do a graceful end for the Active (which will cause the switchover to Standby)

Status at start of this step:
host1: Active
host2: Standby
master file locked by: host1
active file locked by: host1
standby file locked by: host2

Now let's look again at the traces from host1 when the Active is terminating when using the command:
endmqm -is QMMI1

The flag -is means:
-i => immediate
-s => notify reconnectable clients to reconnect.

File: AMQ30939.0.EC.amqzxma0.host1.FMT

Note:
The traces do NOT show the explicit unlocking of the files.

During the ending process, the Active queue manager checks that the standby file is locked (which means that there is a standby) and then proceeds to notify the reconnectable clients to reconnect:

```
 12:21:53.852339   30939.16       :     --{  rrxNotifyReconnectableClients
 12:21:53.852348   30939.16       :     ---{  rppConnectPool
…
 12:21:55.359398   30939.1        :     --{  zutReleaseQMFileLocks
 12:21:55.359402   30939.1        :        Flags(0X21)
 12:21:55.359405   30939.1        :     ---{  xcsCloseFileLock
 12:21:55.360284   30939.1        :        Data: 0x0000000e 0x00000000
 12:21:55.360968   30939.1        :     ---}  xcsCloseFileLock rc=OK FunctionTime=1563
 12:21:55.360980   30939.1        :     ---{  xcsCloseFileLock
 12:21:55.361311   30939.1        :        Data: 0x0000000d 0x00000000
 12:21:55.361785   30939.1        :     ---}  xcsCloseFileLock rc=OK FunctionTime=805
 12:21:55.361797   30939.1        :     --}  zutReleaseQMFileLocks rc=OK
FunctionTime=2399
```

At this point, the Active instance has released the locks, and the master/active files are ready to be locked by the Standby.

directory: /mqha/qmgrs/QMMI1/errors
File: AMQERR01.LOG

The error logs now indicate that the previously active instance in host1 (riggioni1) is now ended:

04/23/2023 12:21:55 PM - Process(30939.1) User(mqm) Program(amqzxma0)
Host(riggioni1.fyre.ibm.com) Installation(Installation1)
VRMF(9.3.2.0) QMgr(QMMI1)
Time(2023-04-23T19:21:55.050Z)
CommentInsert3(QMMI1)
AMQ8004I: IBM MQ queue manager 'QMMI1' ended.
EXPLANATION:
IBM MQ queue manager 'QMMI1' ended.

+ Step 5-b: Because the Standby is querying every 2 seconds to see if it can get the master lock, then it very quickly gets it:

Status at start of this step:
host1: no instance running
host2: Standby
master file locked by: none
active file locked by: none
standby file locked by: host2

The following is from the trace of the EC from host2:
AMQ118435.0.EC.amqzxma0.host2.FMT

The Active instance has unlocked the master file, and now the Standby queries for the file and the file has no longer exclusive locks and the Standby gets the locks (becoming an Active instance)

The Standby got the lock for the file and now writes its data into the file:

```
12:21:47.518024   118435.1         :          ThreadEventName: hevtEC, ThreadEventId: 1
*12:21:55.362991   118435.14        :      ----}  xcsWaitFileLock rc=OK
FunctionTime=8093123
 12:21:55.363068   118435.14        :          fd(21)
 12:21:55.363073   118435.14        :      ---}  xcsOpenFileLock rc=OK
FunctionTime=8093105
 12:21:55.363081   118435.14        :          Lock details to write into master:
suvereto1.fyre.ibm.com,7225327562557880603,1682277667,501,501
 12:21:55.363086   118435.14        :      ---{  xcsWriteFileLock
 12:21:55.363090   118435.14        :
LockData(suvereto1.fyre.ibm.com,7225327562557880603,1682277667,501,501)
 12:21:55.375698   118435.14        :      ---}  xcsWriteFileLock rc=OK FunctionTime=12612
```

It does the same with the active file:

```
12:21:55.375772   118435.14        :      ---{  xcsOpenFileLock
 12:21:55.375791   118435.14        :          FileName(/mqha/qmgrs/QMMI1/active)
LockMode(0X1)
 12:21:55.377441   118435.14        :      ----{  xcsWaitFileLock
 12:21:55.377477   118435.14        :          fd(22) LockType(0X1) timeout(-1)
 12:21:55.378063   118435.14        :      ----}  xcsWaitFileLock rc=OK FunctionTime=622
 12:21:55.378079   118435.14        :          fd(22)
 12:21:55.378083   118435.14        :      ---}  xcsOpenFileLock rc=OK FunctionTime=2311
 12:21:55.378090   118435.14        :          Lock details to write into active:
suvereto1.fyre.ibm.com,7225327562557880603,1682277667,501,501
 12:21:55.378094   118435.14        :      ---{  xcsWriteFileLock
```

```
 12:21:55.378099   118435.14        :
LockData(suvereto1.fyre.ibm.com,7225327562557880603,1682277667,501,501)
 12:21:55.391067   118435.14        :       ---}  xcsWriteFileLock rc=OK FunctionTime=12973
```

Then releases the lock for the standby file

```
 12:21:55.391136   118435.14        :       ---{  xcsCloseFileLock
 12:21:55.391761   118435.14        :           Data: 0x00000016 0x00000000
 12:21:55.392183   118435.14        :       ---}  xcsCloseFileLock rc=OK FunctionTime=1047
```

+ Step 6: Status after switchover

Now the Standby instance in host2 has become the Active instance.

Status at start of this step:
host1: no instance running
host2: Active
master file locked by: host2
active file locked by: host2
standby file locked by: none

Display the status of the queue managers after the switchover has completed

```
  Host2:  dspmq -xf -m QMMI1
QMNAME(QMMI1)                                   STATUS(Running elsewhere)
    INSTANCE(suvereto1.fyre.ibm.com) MODE(Active)
        master(suvereto1.fyre.ibm.com,7225327562557880603)
        active(suvereto1.fyre.ibm.com,7225327562557880603)
```

Trace file:
AMQ31123.0.dspmq.2.host1.FMT

```
 12:22:24.824996    31123.1         :      ---{  xcsQueryFileLock
 12:22:24.824998    31123.1         :          FileName(/mqha/qmgrs/QMMI1/master)
 12:22:24.826601    31123.1         :          Lock data read ()
 12:22:24.826616    31123.1         :          Data: 0x00000008 0x00000000
 12:22:24.826857    31123.1         :      ---}  xcsQueryFileLock rc=OK FunctionTime=1861
 12:22:24.826883    31123.1         :          Lock data for /mqha/qmgrs/QMMI1/master is
<suvereto1.fyre.ibm.com> <LOCKED>
…
 12:22:24.826899    31123.1         :      ---{  xcsQueryFileLock
 12:22:24.826902    31123.1         :          FileName(/mqha/qmgrs/QMMI1/standby)
 12:22:24.828171    31123.1         :          Lock data read ()
 12:22:24.828186    31123.1         :          Data: 0x00000008 0x00000000
 12:22:24.828456    31123.1         :      ---}  xcsQueryFileLock rc=OK FunctionTime=1557
 12:22:24.828468    31123.1         :          Lock data for /mqha/qmgrs/QMMI1/standby is
<suvereto1.fyre.ibm.com> <NOT LOCKED>
 12:22:24.828471    31123.1         :      --}  zutQueryQMFileLocks rc=OK FunctionTime=1577
```

Let's look at the error log for the message that indicates that the standby from host2 Host(suvereto1) is now the Active

04/23/2023 12:21:55 PM - Process(118435.1) User(mqm) Program(amqzxma0)
Host(suvereto1.fyre.ibm.com) Installation(Installation2)
VRMF(9.3.2.0) QMgr(QMMI1)
Time(2023-04-23T19:21:55.401Z)
CommentInsert3(QMMI1)
AMQ8352I: IBM MQ queue manager 'QMMI1' becoming the active instance.
EXPLANATION:
The standby instance of queue manager instance 'QMMI1' is becoming the active instance.

+ Step 7: Display the status of the queue managers after the switchover has completed

```
  host1:  dspmq -xf -m QMMI1
mqm@host1: /var/mqm/trace
$ dspmq -xf -m QMMI1
QMNAME(QMMI1)                                             STATUS(Running)
    INSTANCE(suvereto1.fyre.ibm.com) MODE(Active)
        master(suvereto1.fyre.ibm.com,7225327562557880603)
        active(suvereto1.fyre.ibm.com,7225327562557880603)

  host2:  dspmq -xf -m QMMI1
mqm@host2: /var/mqm/trace
$ dspmq -xf -m QMMI1
QMNAME(QMMI1)                                             STATUS(Running)
    INSTANCE(suvereto1.fyre.ibm.com) MODE(Active)
        master(suvereto1.fyre.ibm.com,7225327562557880603)
        active(suvereto1.fyre.ibm.com,7225327562557880603)

Trace file:
AMQ118608.0.dspmq.2.host2.FMT

 12:22:48.431553   118608.1         :      --{  xcsQueryFileLock
 12:22:48.431559   118608.1         :         FileName(/mqha/qmgrs/QMMI1/master)
 12:22:48.434609   118608.1         :         Lock data read ()
 12:22:48.434659   118608.1         :         Data: 0x00000008 0x00000000
 12:22:48.434683   118608.1         :      --}  xcsQueryFileLock rc=OK FunctionTime=3130
 12:22:48.434708   118608.1         :         Lock data for /mqha/qmgrs/QMMI1/master
is <suvereto1.fyre.ibm.com> <LOCKED>

12:22:48.434734   118608.1         :      --{  xcsQueryFileLock
 12:22:48.434737   118608.1         :         FileName(/mqha/qmgrs/QMMI1/active)
 12:22:48.435835   118608.1         :         Lock data read ()
 12:22:48.435863   118608.1         :         Data: 0x00000008 0x00000000
 12:22:48.436296   118608.1         :      --}  xcsQueryFileLock rc=OK FunctionTime=1562
 12:22:48.436325   118608.1         :         Lock data for /mqha/qmgrs/QMMI1/active is
<suvereto1.fyre.ibm.com> <LOCKED>

 12:22:48.436347   118608.1         :      --{  xcsQueryFileLock
 12:22:48.436351   118608.1         :         FileName(/mqha/qmgrs/QMMI1/standby)
 12:22:48.437756   118608.1         :         Lock data read ()
 12:22:48.437779   118608.1         :         Data: 0x00000008 0x00000000
 12:22:48.438443   118608.1         :      --}  xcsQueryFileLock rc=OK FunctionTime=2096
 12:22:48.438464   118608.1         :         Lock data for /mqha/qmgrs/QMMI1/standby
is <suvereto1.fyre.ibm.com> <NOT LOCKED>

+++ end +++
```