# IBM MQ Queue Manager Clusters & Disaster Recovery Basics

6HVY\_Andy\_Emmett  
Published on 15/02/2019*/ Updated on 18/02/2019*

# Overview

In this blog article, myself along with colleague Sean Cawood, describe the basics of the IBM MQ inbuilt disaster recovery capabilities for Queue Manager Clusters. After reading the article, users will have a basic understanding of how a failed server hosting a queue manager in a cluster (which is no longer accessible), can be replaced with a second instance of the queue manager. Our goal is for this to serve as the foundation for more in-depth discussions on the practices for disaster recovery in the context of Queue Manager Clusters.

# Scope : What is and isn’t covered

It’s important to say up front, that the target audience for this series of posts, is primarily those with a working understanding of Queue Manager Clusters.

Here is a list of what is covered in this article …

1. Introduction and description of our example configuration
2. Reminder : the differences between Disaster Recovery (DR) and High Availability (HA)
3. Discuss HA from a Queue Manager Cluster perspective
4. Discuss basic DR capabilities for Queue Manager Clusters

# Introduction

In the diagram below, we have an example cluster **(CLUSTER1)** which we use for our description. In the cluster is a queue manager named LONDON where applications can connect and put messages to queues in the cluster. One such application puts messages to a named queue of which there are six instances, hosted on six different queue managers in the configuration. Typically, the messages would be routed to these queue managers using the standard [workload management algorithm](https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_9.0.0/com.ibm.mq.ref.con.doc/q082390_.html) and perhaps evenly distributed among them in a round-robin fashion. In our example, we will focus on the failure of the LONDON queue manager, though to be more specific, the failure of the server which hosts the LONDON queue manager making it inaccessible.

A picture containing table, man, hanging, various

Description automatically generated

Fig 1. Example cluster configuration.

# Helpful Reminder

There is a good chance that you may have read what we are about to say elsewhere, for example, [Integrating the IBM MQ Appliance into your IBM MQ Infrastructure](http://www.redbooks.ibm.com/abstracts/sg248283.html) (see chapter 10). However, it is worth saying it again here …

The focus for high availability (often referred to simply as HA) is usually based on single components within the whole system. High availability designs usually aim to provide a rapid takeover of a failed system (including the data) by another equivalent system. The data that is on a disk might be owned temporarily by one server (which exhibits a failure) and later taken over by another server that also is attached to the same disk where the business data is stored.

On the other hand, disaster recovery (DR), is typically the process and procedures that are used to restore business continuity in the time soon after an event that caused significant disruption to normal service. The event could be the result of natural causes or as a direct result of some human intervention. The aim of disaster recovery, is to restore services that are needed for continued business activity. Within a disaster situation, it can be assumed that there can be some limited loss of data and a time for which services can be unavailable. Perhaps unexpectedly, this time can be several hours and not only minutes.

Typically, within the DR pattern, there is a plan for the business IT infrastructure. This plan might consist of an ordered list of infrastructure resources to be brought online for the most important services to provide the minimal system. This plan is then built upon, bringing up secondary, tertiary services, and so on, until a complete IT environment is available which functions as a whole to provide all services that the business needs.

# Queue Manager Clusters & High Availability

Our diagram shows that queue manager clusters can have a form of high availability, by using a collection of resources, which do not share disks. Each of them define and expose a queue in the cluster. Every other member of the cluster can (if required) see them all. The queue managers either store or immediately give received messages, directly to connected applications. Should a failure with one of the queue managers other that LONDON occur, then LONDON would recognize this. For example, the channel to the failed queue manager, may go into retry state. There may be a small delay to LONDON knowing this, but when it does, LONDON can behave accordingly by preventing (perhaps temporarily) sending messages to the failed queue manager. In this case, there are effectively five possible destinations to choose from as far as LONDON is concerned.

A close up of a device

Description automatically generated

Fig 2. Server failure where a cluster queue is hosted.

#### **IBM MQ Appliance**

The above description changes, if in the scenario we discuss, a pair on IBM MQ Appliances are employed to provide additional HA capabilities. Now, in the new diagram (below), should one of the pair of appliances suffer a failure, ownership of the queue manager(s) will be taken over by the second appliance, in fairly short order. This means that LONDON will still be able to use all six instances of the application queue in the cluster.

A picture containing table, various, man, items

Description automatically generated

Fig 3. Failure of a single IBM MQ Appliance configured as a High Availability pair.

# Queue Manager Clusters & Disaster Recovery

The focus of this article is disaster recovery and continuing with our same example, we will consider what happens should LONDON fail. Remember that we said LONDON was the only queue manager to which our application connects to send messages to the six remote queue managers? Well, if LONDON fails, then no more messages can be put or routed. The remote queue managers can only process messages that have already reached them. This could be a big problem. Possibly, no commands can be issued against the LONDON queue manager and perhaps there are now messages stranded on the transmission queues.

A picture containing man, table

Description automatically generated

Fig 4. Failed Queue Manager in a cluster configuration.

## **How does IBM MQ help ?**

IBM MQ cannot solve a disk crash for you, nor a failed network interface card. What it can do, is ease the ability to bring a replacement queue manager with the same name, into the configuration, and automatically notify every member that has registered interest in this named queue manager, of the replacement. Specifically, that the new queue manager identifier, replaces the old queue manager identifier, from this point in time onward.

A close up of a device

Description automatically generated

Fig 5. Introducing a 2nd queue manager instance.

## **What makes this possible? – Uniqueness**

A queue manager participating in a cluster, maintains a collection of records about itself and other queue managers along with records for other object types (the cluster cache). The queue manager type records (typically referred to as Cluster Queue Manager records) are distinct from each other using a number of markers…

* CLUSRCVR Channel Name, simply the name that was attributed to the channel when it was defined.
* Queue Manager Identifier (QMID) uniquely identifies a given single instance of queue manager.
* Cluster(s) Each CLUSRCVR channel can be shared a single cluster (using the CLUSTER attribute of the channel) or multiple clusters using the CLUSNL attribute and making use of a NAMELIST

### **Channel Name**

This is the primary aspect of a unique cluster queue manager. The channel name together with the queue manager identifier, will logically form a new record if one does not already exist in the local cache for the combination. The channel name originates from the cluster receiver channels (CLUSRCVR) defined on queue managers in the configuration.

### **Queue Manager Identifier**

Each queue manager when it is created, is given a identifier. This QMID ensures (within the boundaries of a single organization) that queue managers are uniquely identifiable. It is therefore possible to have two functioning queue managers with the same name, participating in a cluster completely independently of each other even if they have the same channel name for their respective CLUSRCVR channels. Please note, this is strongly discouraged and we advise users to have unique queue manager names in their cluster configurations.  
Here are two examples of the QMID formats…

* LONDON\_2018-07-16\_16.20.54 (Distributed Platforms)
* MYQM.D4B481EFF1C1BC26 (z/OS)

To find the QMID of a queue manager on distributed platforms, a user with appropriate authority, can use MQSC commands when the queue manager is running.

A screenshot of a cell phone

Description automatically generated

Fig 6. Example of queue manager LONDON’s QMID using Windows(TM).

### **Other important record information**

Along with the above unique fields, a cluster queue manager record also has two other important pieces of information …

* An ‘In Use’ flag. A marker which indicates that this record the active instance of a queue manager with the same combination (where there is more than 1) of queue manager name, cluster(s) and channel name.
* A Sequence Number. This is typically an ever increasing value to track changes to the record.

# Replacing a failed Cluster Queue Manager

As discussed earlier, it is possible (though not advised) to have two queue managers with the same name in a cluster configuration. The only way in which they can be considered as two distinct queue managers, is if they both have unique CLUSRCVR channel names. For example, if we wanted there to be two LONDON queue managers, we would need something like …

* QM : LONDON (1) / CLUSRCVR : TO.LONDON.TOWN
* QM : LONDON (2) / CLUSRCVR : TO.LONDON.CITY

However, in our considered example, we only have one queue manager called LONDON (now failed) and for the purpose of this example, lets assume its CLUSRCVR channel name is TO.LONDON  
In order to replace this queue manager we do the following …

1. Create a new queue manager called LONDON : crtmqm LONDON
2. Start the new queue manager : strmqm LONDON
3. Configure the queue manager with definitions that match the old queue manager, specifically, ensuring that the CLUSRCVR channel name is the same, TO.LONDON

A screenshot of a cell phone

Description automatically generated

Fig 7. New LONDON queue manager with a different identifier or UUID.

## **The Main Point**

**In order to replace a failed queue manager in a cluster configuration, a user simply needs to reconstruct it with the same name. The cluster receiver channels must also have the same names as those defined on the original queue manager. It is then introduced into the cluster using the sender channels (which would have been defined during the reconstruction). When this new instance contacts a full repository (regardless of the new instance being a partial repository or a full repository itself), it will trigger updates to be pushed throughout the cluster. These updates indicate to the other participants that the queue manager with the specific name (in our case LONDON), now has a new QMID. At this time, the original instance has its ‘In Use’ flag removed from its record and as such is no longer the active version of LONDON. Meanwhile, the new instance is marked as being the new ‘In Use’ version of LONDON.**

Please note :

* In this article we have discussed replacing a queue manager with another new instance of itself. To be clear, we have not been discussing restoring the failed queue manager from a backup. If we did restore from a backup, then there would only ever be one QMID for the named queue manager in the cluster configuration. However, this could lead to problems if all of the other members of the cluster have records for the queue manager, from an earlier point in time. This subject matter would require a whole other article of its own to do it justice.
* Configuration of the channel hostnames may need to be different than those used for the original definition of LONDON
* If we were to create the new instance with a different name for the cluster receiver channel, two instances of this queue manager would exist. This could lead to confusion over which version of the queue manager with the name LONDON, messages would be routed to when they are put by other members of the cluster.

A close up of a logo

Description automatically generated

Fig 8. Conflicting instances of LONDON queue manager.

# Forgetting The Original Queue Manager Instance

So to continue with the scenario of having introduced a new instance of LONDON into the cluster and is now being used in place of the original by all members, there is remaining knowledge of the original LONDON queue manager (though now not in use) in the configuration (inside other cluster members). This information may show up when running some cluster commands, for example, ‘dis qcluster’.

### **Finding the QMID of the old LONDON instance**

On one of the full repository queue managers for the cluster, we want to issue ‘dis clusqmgr’. This will report all of the definitions (effectively all CLUSRCVR channels) that are known. This will include the original LONDON queue manager and my look something like the following …

C:\>runmqsc FR1

5724-H72 (C) Copyright IBM Corp. 1994, 2015.

Starting MQSC for queue manager FR1

dis clusqmgr(\*) qmid

3 : dis clusqmgr(\*) qmid

AMQ8441I: Display Cluster Queue Manager details.

CLUSQMGR(ANDY) CHANNEL(TO.ANDY)

CLUSTER(TEST) QMID(ANDY\_2019-02-13\_06.17.35)

AMQ8441I: Display Cluster Queue Manager details.

CLUSQMGR(FR1) CHANNEL(TO.FR1)

CLUSTER(TEST) QMID(FR1\_2019-02-13\_06.23.28)

AMQ8441I: Display Cluster Queue Manager details.

CLUSQMGR(FR2) CHANNEL(TO.FR2)

CLUSTER(TEST) QMID(FR2\_2019-02-13\_06.23.33)

AMQ8441I: Display Cluster Queue Manager details.

CLUSQMGR(LONDON) CHANNEL(TO.LONDON)

CLUSTER(TEST) **QMID(LONDON\_2018-07-16\_16.20.54)**

AMQ8441I: Display Cluster Queue Manager details.

CLUSQMGR(LONDON) CHANNEL(TO.LONDON)

CLUSTER(TEST) QMID(LONDON\_2019-02-12\_13.49.47)

Fig 9. MQSC output showing two instance of LONDON.

### **Removing the old instance of LONDON**

We can continue to use the same full repository to issue the next command which will be RESET CLUSTER, or we could issue it against the other full repository (assuming that there are two, the recommended number of repositories for a given cluster).

Please note:

* The information regarding the old LONDON instance, would be removed naturally from the entire cluster configuration if reset cluster were not used and as long as the queue manager is not brought back into existence This would happen after no more than 90 days. RESET CLUSTER is used to cleanup this information sooner. Click [here](https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_7.5.0/com.ibm.mq.ref.con.doc/q082370_.htm) for further details about the RESET CLUSTER command.

C:\>runmqsc FR1

5724-H72 (C) Copyright IBM Corp. 1994, 2015.

Starting MQSC for queue manager FR1.

reset cluster(TEST) ACTION(FORCEREMOVE) QMID('LONDON\_2018-07-16\_16.20.54') QUEUES(YES)

1 : reset cluster(TEST) ACTION(FORCEREMOVE) QMID('LONDON\_2018-07-16\_16.20.54') QUEUES(YES)

AMQ8559: RESET CLUSTER accepted.

Fig 10. MQSC output showing single LONDON instance after RESET CLUSTER.

# Summary

So, in this short article, we have described how to introduce a replacement cluster queue manager for one which has failed. This is essentially the basics regarding what would typically be considered, an unprepared disaster scenario. Ideally, to be more effective for a disaster scenario, it would be advantageous to be prepared. This might mean, having servers installed and ready to go and / or in some kind of standby mode, ready to take over the workload from a queue manager that has ‘failed’ or is offline for maintenance for example.  
Describing and testing these kinds of scenarios requires a level of detail beyond the scope of this article, and we endeavor to bring you something more on this topic. In the meantime, we hope this has been useful.

**TAGS**[**CLUSTERING**](https://developer.ibm.com/messaging/tag/clustering/)**,**[**DISASTER RECOVERY**](https://developer.ibm.com/messaging/tag/disaster-recovery/)

*by*[*6HVY\_Andy\_Emmett*](https://developer.ibm.com/messaging/author/aemmett/)

4 comments on"IBM MQ Queue Manager Clusters & Disaster Recovery Basics"

1. **6HVY\_Andy\_Emmett** July 25, 2019

Hello,  
It seems that this is not the correct article for the question. This post is specifically about the in built disaster recovery mechanisms for Queue Manager Clusters. Could you elaborate on the question? Do you mean high availability in terms of something like MSCS / RDQM ? With more details perhaps we can find the correct post for the question.

Regards Andy Emmett

1. **Anonymous** July 25, 2019

what are the prerequsites to configure the MQ HA cluster setup ?

1. **6HVY\_Andy\_Emmett** April 30, 2019

Hello Florian.  
It is always very useful to receive comments on the articles. Thanks very much for your feedback.

1. **FLORIAN MANSCHWETUS** April 30, 2019

Very informative and helpful article, bringing a lot of light into a quite difficult and pitfall prone area of MQ.