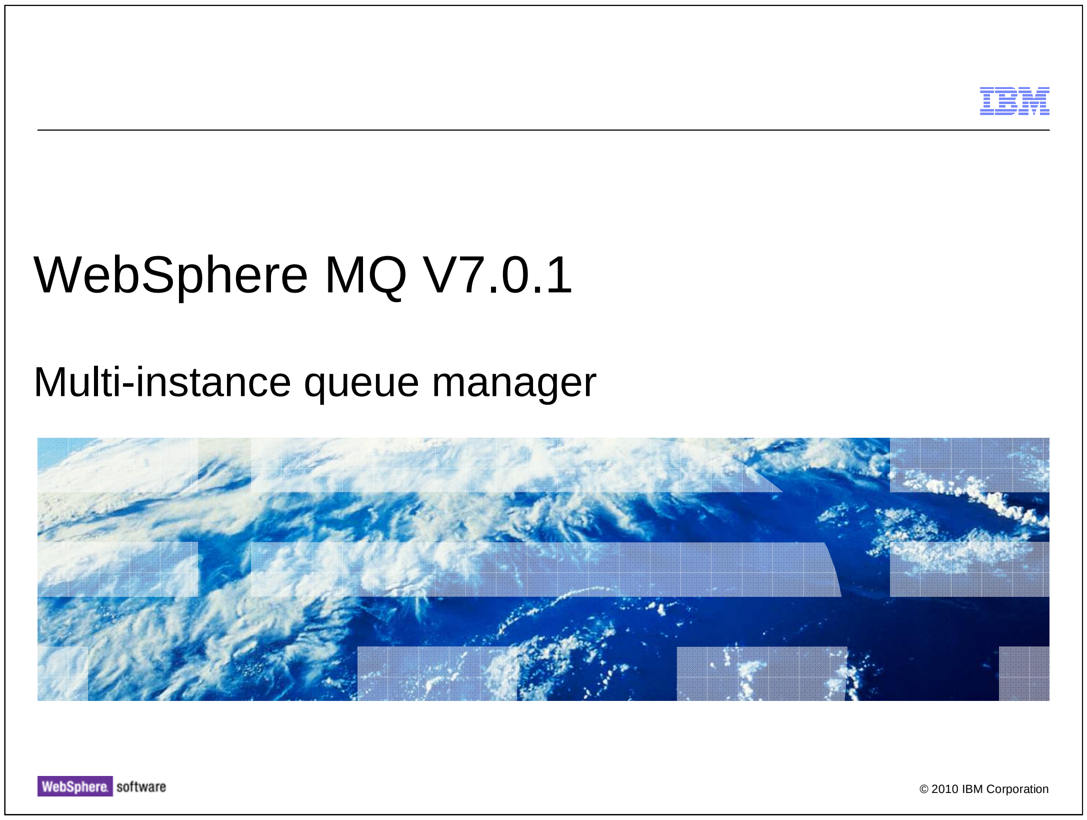


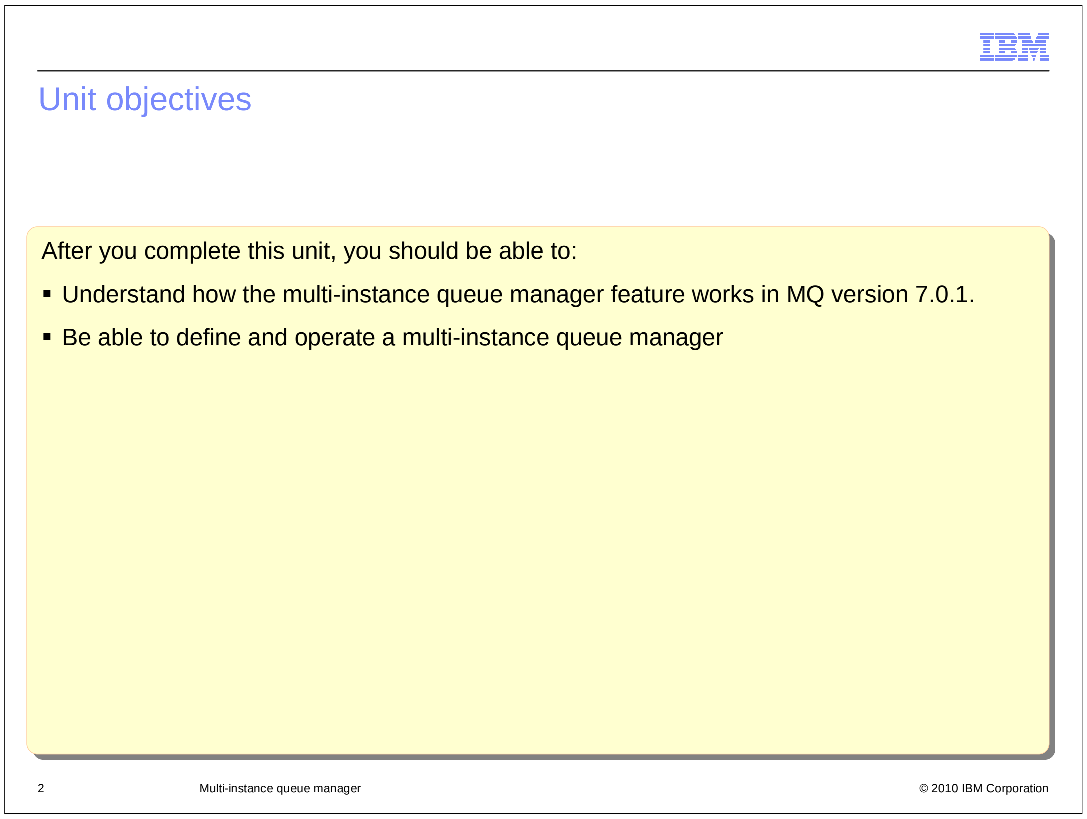
WebSphere MQ V7.0.1

Multi-instance queue manager

WebSphere software

This unit explains the multi-instance queue manger feature introduced in WebSphere® MQ version 7.0.1.

This unit assumes a reasonable understanding of how WebSphere MQ works.



Unit objectives

After you complete this unit, you should be able to:

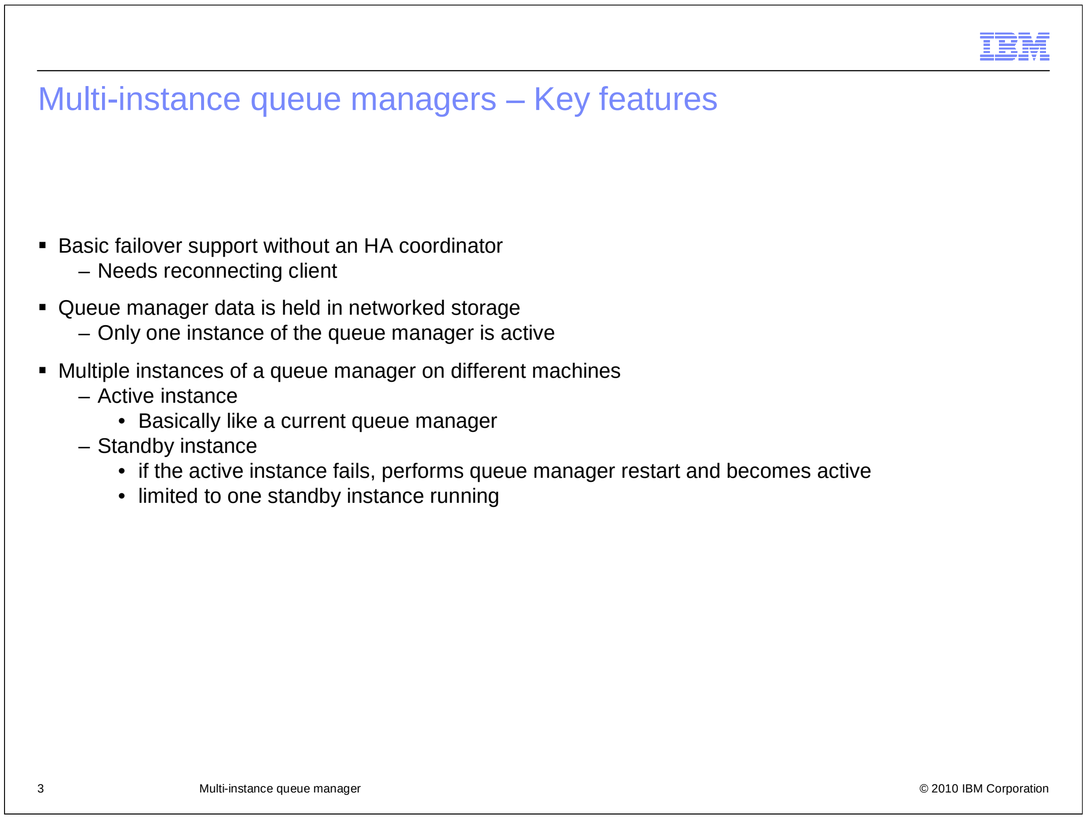
- Understand how the multi-instance queue manager feature works in MQ version 7.0.1.
- Be able to define and operate a multi-instance queue manager

2 Multi-instance queue manager © 2010 IBM Corporation

After you complete this unit you should have some understanding of how existing applications using the queued publish / subscribe mechanism is migrated to MQ version 7.

You will also be aware of some of the issues that need to be considered before migration.

This unit does not attempt to cover the full range of syntax and options available, for which you should refer to the product information center.



Multi-instance queue managers – Key features

- Basic failover support without an HA coordinator
  - Needs reconnecting client
- Queue manager data is held in networked storage
  - Only one instance of the queue manager is active
- Multiple instances of a queue manager on different machines
  - Active instance
    - Basically like a current queue manager
  - Standby instance
    - if the active instance fails, performs queue manager restart and becomes active
    - limited to one standby instance running

3 Multi-instance queue manager © 2010 IBM Corporation

The multi-instance queue manager feature of WebSphere MQ version 7.0.1 is part of the basic failover support now provided as a base part of the distributed product. Together with the “reconnecting client” feature, which is the subject of a separate presentation, multi instance queue manager provides a basic failover capability without the need of a High Availability coordinator. This means that in the event of the failure of the MQ queue manager, or the machine it is running on, a standby instance will automatically takeover and client connections will be transferred to this new instance.

The key technical feature in achieving this is the placement of the queue manager data, including its logs, in networked storage. This means that all instances of a queue manager can access the same data. The information center gives details of the supported file systems that can be used to host the queue managers data. The essential requirement is that the file system is POSIX compliant; this allows the queue manager instances to lock the files appropriately.

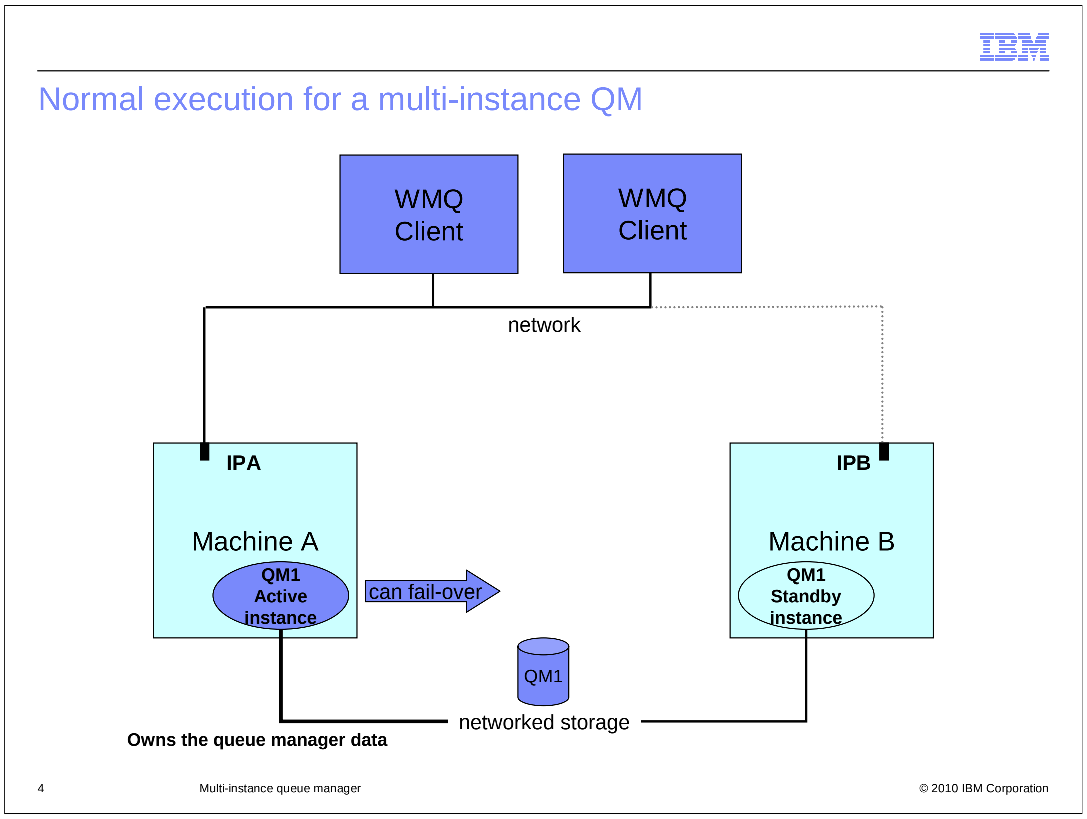
Multiple instances of a queue manager can then be defined on different machines that all have access to the networked file system. It should be noted that the instances must all be running on the same operating systems. It is not possible to have an active system on Windows® with a standby on Linux® for example.

The active instance behaves almost exactly like a “normal” queue manager. It owns the queue manager data and logs and accepts all the connections to client applications and by channels to other queue managers. Some new variations in the stop queue manager commands are now supported.

The standby instance is basically another copy of the queue manager that has begun the start process but has stopped waiting to access the queue manager’s data and logs. In the event of the active instance failing or receiving a switchover command the standby instance acquires the needed file system locks and continues its startup. In the process it will read the logs, backing out any uncommitted transactions. This instance then becomes the active instance and will now accept connections. An additional instance could now be started as the standby.

Note at most one standby instance can be started, although many can be defined. So machines A, B and C may all have definitions of Queue Manager X, A may be the normally active instance, with a standby instance running on B. In the event of a failure of A, B takes over and becomes active. At this point the instance on C could be started and would become a standby instance.

The next slides illustrate the multi-instance queue manager.



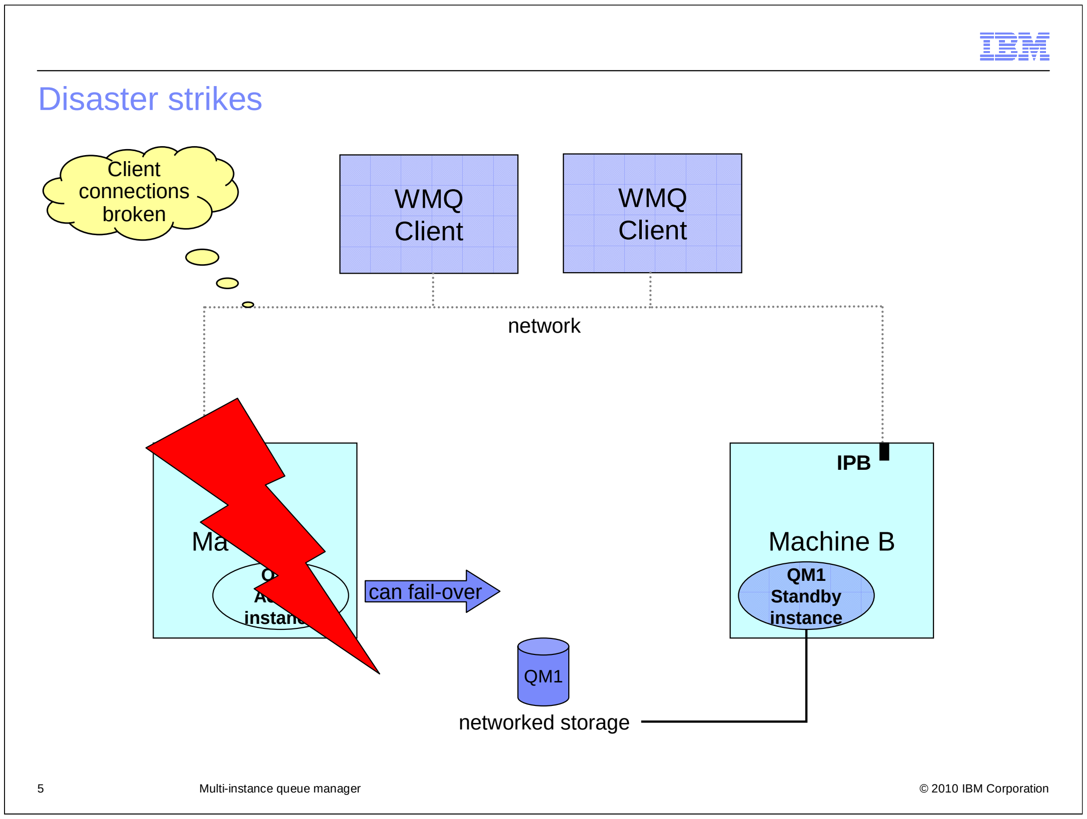
Normal execution for a multi-instance QM

This slide illustrates what might be the normal state of operations for a multi-instance queue manager.

Client applications, which might be MQ API or JMS applications, are connected over an IP network to the queue manager “QM1” which is running on machine A, on the left. All clients are connected to machine A on IP address “IPA”. The queue manager “QM1” data is in the shared storage on disk.

The current active connections are shown as bolder solid lines.

Machine B has a standby instance of “QM1” running, as a standby instance it has no clients connected and does not hold the locks on the queue manager data in the shared file system. Note however that machine B has its own IP address “IPB”, and that all clients “know” that address.



Disaster strikes

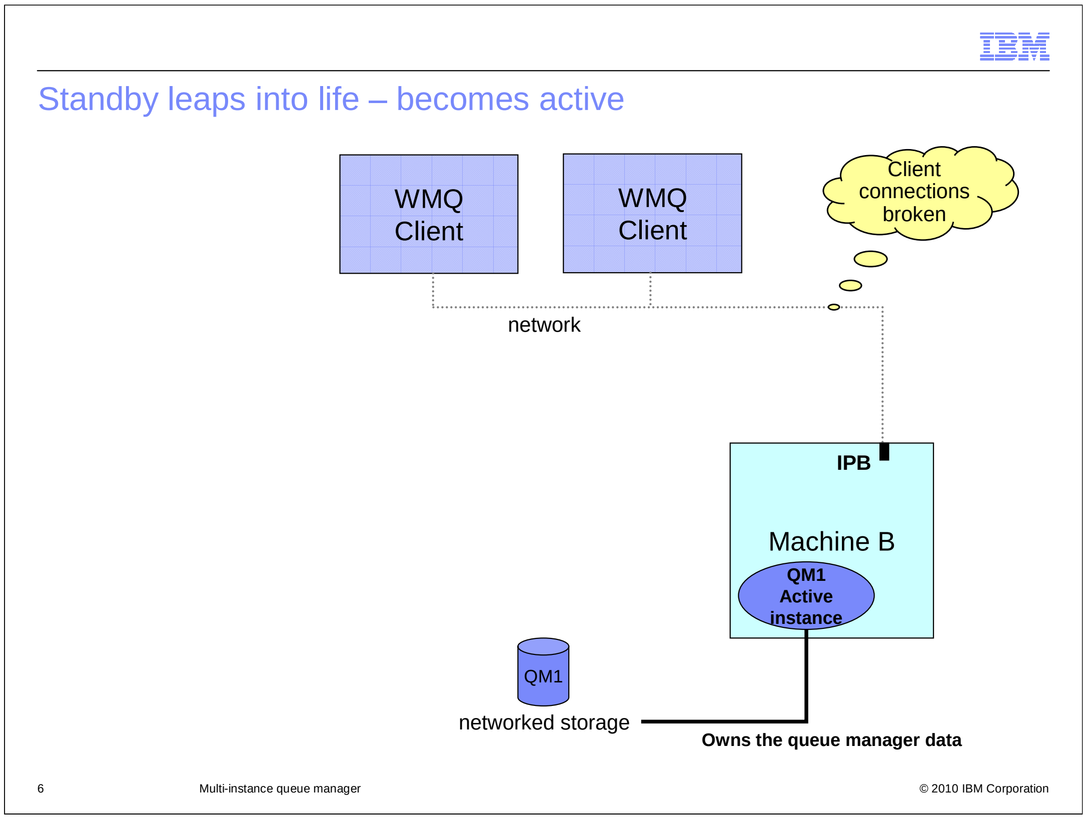
Suppose disaster strikes…

Machine A becomes unavailable. As a consequence one of these two things happen first.

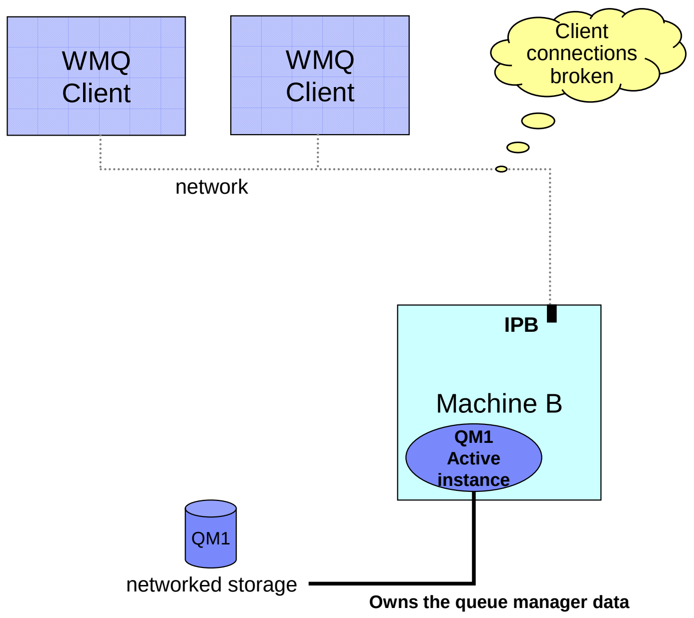
One: The client connections are broken; the MQ code running on the clients will detect this.

Two: The connection to the network storage is broken, and the locks released by the file system. This requirement to release locks sets prerequisites on the shared file systems supported.

This sets the scene for recovery.



Standby leaps into life – becomes active

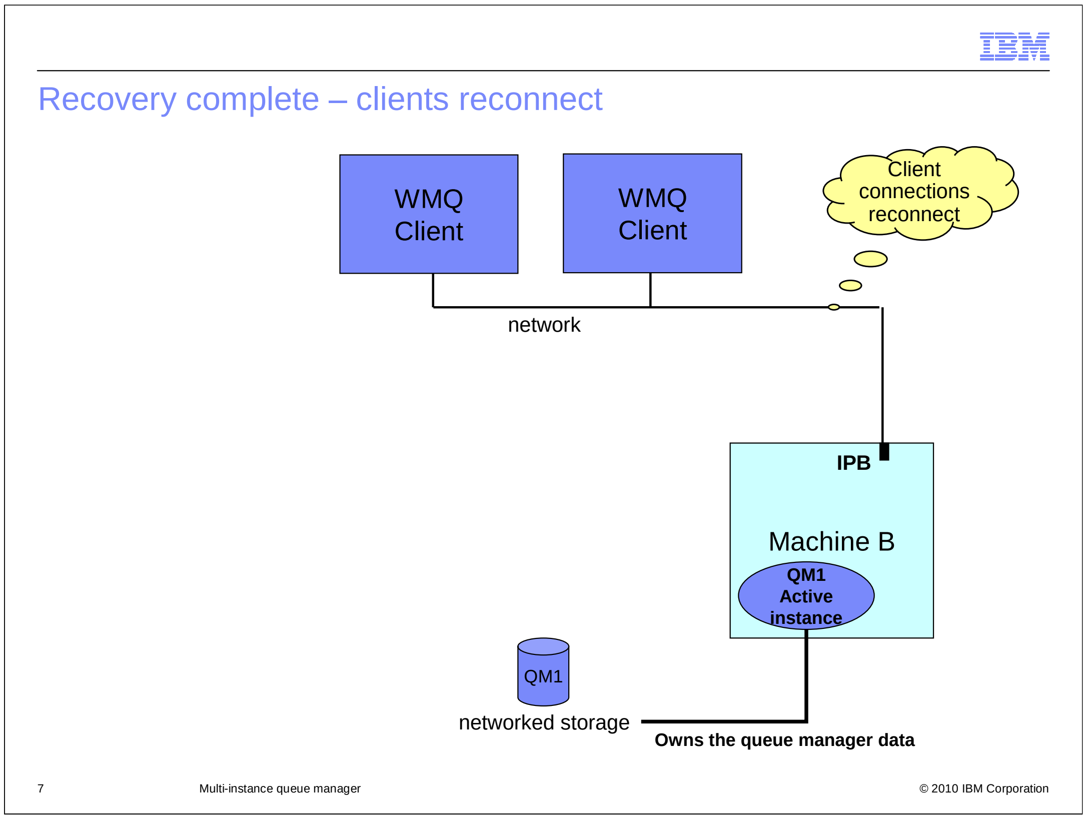


6 Multi-instance queue manager © 2010 IBM Corporation

The first stage of recovery is that the standby instance which was waiting for the locks on the file system acquires the locks and completes its restart.

It goes through a fairly normal startup, processing the log records written by the earlier instance. This means that in flight units of work will be rolled back, persistent messages will survive and non-persistent messages deleted.

The Queue Manager is now fully operational and waiting for clients to connect.



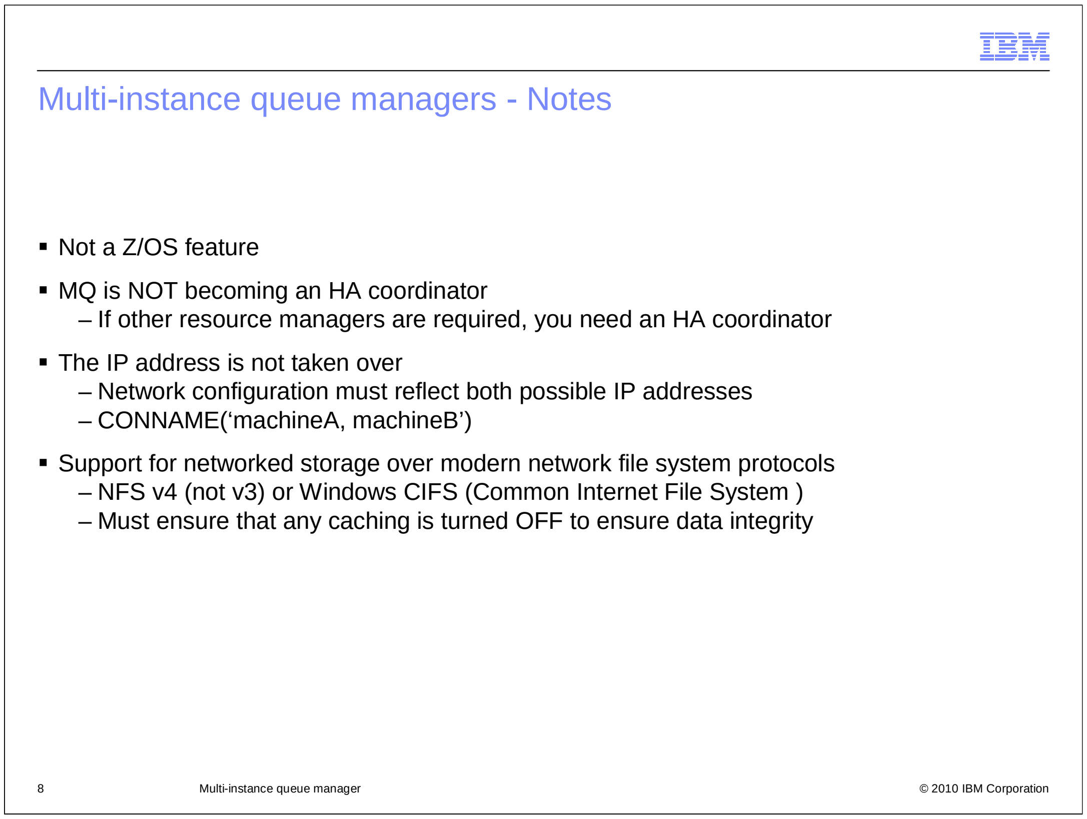
Recovery complete – clients reconnect

7 Multi-instance queue manager © 2010 IBM Corporation

Which is exactly what happens for those clients connected using the reconnecting client feature of MQ 7.0.1.

The client connections that were disconnected from the first instance can now connect the newly active instance. Similarly channels from other connected queue managers will reconnect to this new instance.

At this point a new standby instance could be started on machine C.



Multi-instance queue managers - Notes

- Not a Z/OS feature
- MQ is NOT becoming an HA coordinator
  – If other resource managers are required, you need an HA coordinator
- The IP address is not taken over
  – Network configuration must reflect both possible IP addresses
  – CONNAME(‘machineA, machineB’)
- Support for networked storage over modern network file system protocols
  – NFS v4 (not v3) or Windows CIFS (Common Internet File System)
  – Must ensure that any caching is turned OFF to ensure data integrity

8 Multi-instance queue manager © 2010 IBM Corporation

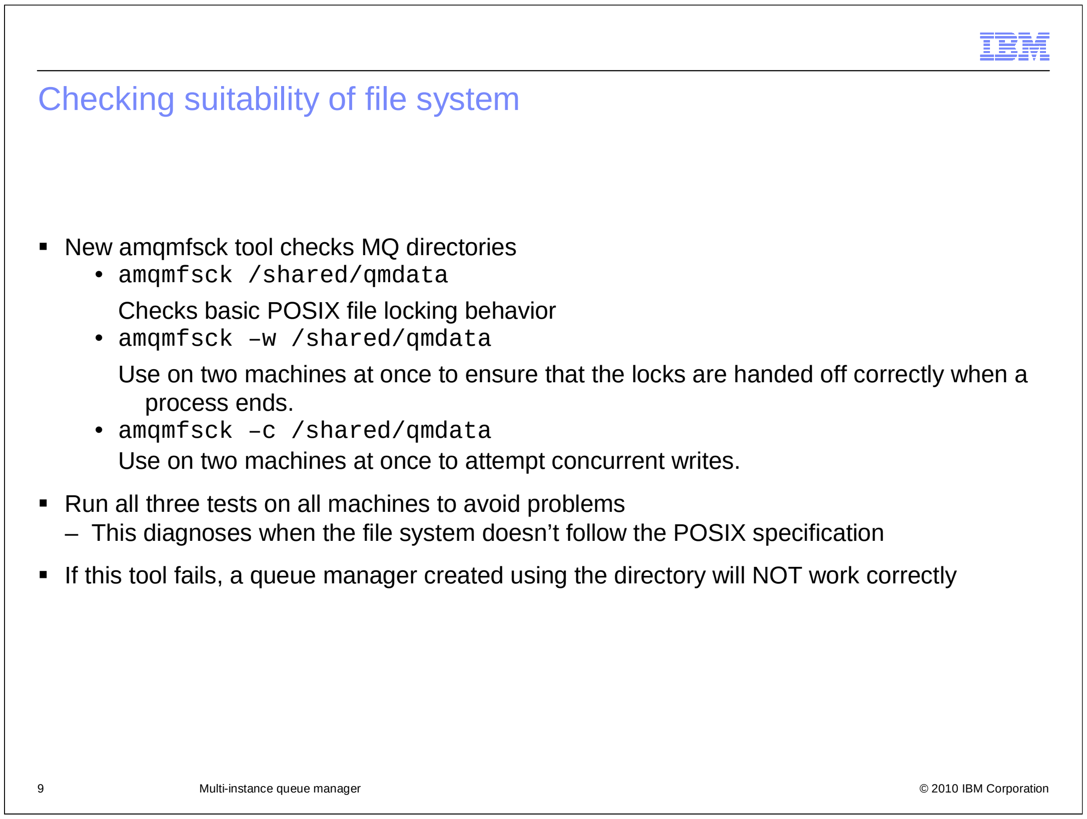
Before looking at the details of configuring multi instance queue managers, here are some notes.

This is a feature of the distributed code, not the Z/OS platform. Z/OS already has a high availability solution available based on Queue Sharing Groups held in sysplex shared memory.

WebSphere MQ is **NOT** becoming a High Availability coordinator. Only MQ resources are passed to the standby instance, if other resource managers, such as data bases, are involved you may need a HA coordinator product.

Again as no HA coordinator is involved there is no IP address takeover. The MQ network configuration must reflect all the possible locations of a queue manager. For this reason the connection name parameter is extended in various places to support a list of IP addresses rather than a single address.

Because WebSphere MQ has strong dependencies on the integrity and locking of the file system only modern network file systems can be supported. In particular the UNIX® Network File System is supported at version 4, but not version 3. In addition any write caching options must be disabled.



Checking suitability of file system

- New amqmfsck tool checks MQ directories
  - `amqmfsck /shared/qmdata`

    Checks basic POSIX file locking behavior
  - `amqmfsck –w /shared/qmdata`

    Use on two machines at once to ensure that the locks are handed off correctly when a process ends.
  - `amqmfsck –c /shared/qmdata`

    Use on two machines at once to attempt concurrent writes.
- Run all three tests on all machines to avoid problems
  - This diagnoses when the file system doesn't follow the POSIX specification
- If this tool fails, a queue manager created using the directory will NOT work correctly

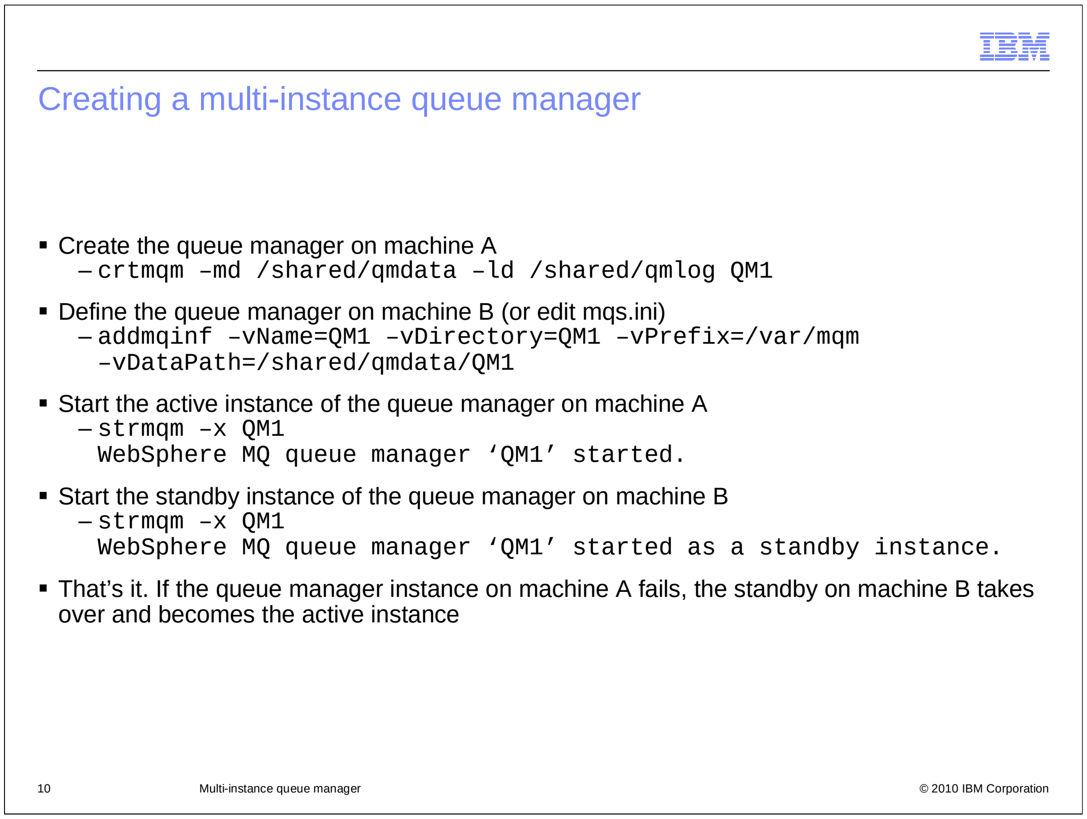
9 Multi-instance queue manager © 2010 IBM Corporation

To help set up a multi-instance queue manager a new tool has been provided to check if a proposed location is suitable for holding the MQ data and log files. This tool should be used before defining a multi-instance queue manager.

The tool should be run on any machine that is planned to run an instance of a queue manager.

The networked file system should be mounted and the user ID that MQ will be running under given access. The AMQMFSCK program then needs to be run three times as indicated; first, to perform a basic POSIX compliance test, then to test locking handover and to test for concurrent writes.

Failure of ANY of these tests indicates that the file system is NOT SUITABLE.



Creating a multi-instance queue manager

- Create the queue manager on machine A
  - `crtmqm –md /shared/qmdata –ld /shared/qmlog QM1`
- Define the queue manager on machine B (or edit mqs.ini)

  -
    ```
    addmqinf –vName=QM1 –vDirectory=QM1 –vPrefix=/var/mqm
    –vDataPath=/shared/qmdata/QM1
    ```
- Start the active instance of the queue manager on machine A

  -
    ```
    strmqm –x QM1
    WebSphere MQ queue manager ‘QM1’ started.
    ```
- Start the standby instance of the queue manager on machine B

  -
    ```
    strmqm –x QM1
    WebSphere MQ queue manager ‘QM1’ started as a standby instance.
    ```
- That’s it. If the queue manager instance on machine A fails, the standby on machine B takes over and becomes the active instance

10 Multi-instance queue manager © 2010 IBM Corporation

Creating the multi-instance queue manager is an asymmetric process. The queue manager is actually created by running commands on one machine to create all the artifacts, the other machines where instances run need to be updated with the location of the queue manager data.

On machine A run a normal crtmqm command to create the queue manager. This command is extended with parameters –md and –ld, which specify the locations for the queue manager data and logs.

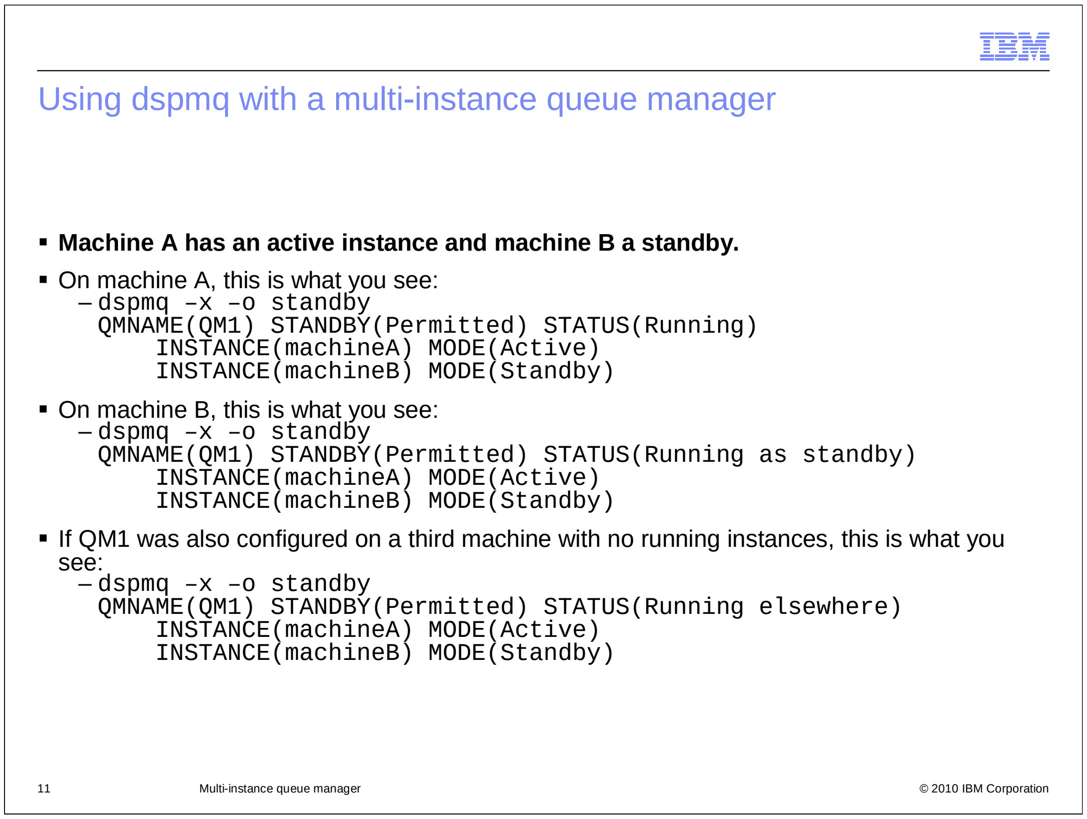
For subsequent instances the “addmqinf@ command can be used to add the information to the mqs.ini data identifying the location of the shared data.

That completes the definition of a multi-instance queue manager.

The two instances are then started using the “strmqm” command with the new “-x” option which indicates that a standby instance is permitted. Note that exactly the same start command is used for an active or standby instance.

The first instance will start as active and report that fact; the second will become a standby instance and report that fact.

You now have a queue manager running as an active/standby pair.



Using dspmq with a multi-instance queue manager

- **Machine A has an active instance and machine B a standby.**
- On machine A, this is what you see:

```
– dspmq –x –o standby
  QMNAME(QM1) STANDBY(Permitted) STATUS(Running)
      INSTANCE(machineA) MODE(Active)
      INSTANCE(machineB) MODE(Standby)
```

- On machine B, this is what you see:

```
– dspmq –x –o standby
  QMNAME(QM1) STANDBY(Permitted) STATUS(Running as standby)
      INSTANCE(machineA) MODE(Active)
      INSTANCE(machineB) MODE(Standby)
```

- If QM1 was also configured on a third machine with no running instances, this is what you see:

```
– dspmq –x –o standby
  QMNAME(QM1) STANDBY(Permitted) STATUS(Running elsewhere)
      INSTANCE(machineA) MODE(Active)
      INSTANCE(machineB) MODE(Standby)
```

11 Multi-instance queue manager © 2010 IBM Corporation

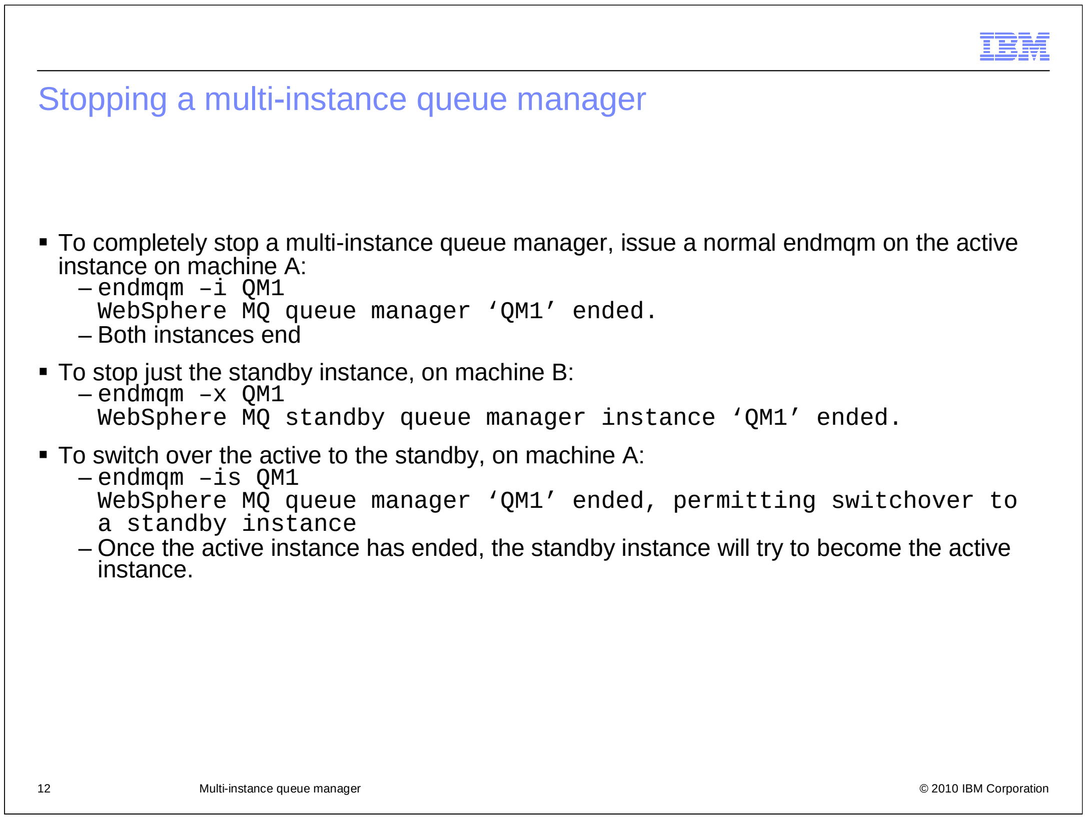
The Display MQ command, dspmq, has been extended to report whether standby instances are permitted, and if so, where they are running.

Suppose machine A is running the active instance and machine B the standby while machine C has the instance defined but no instance is started.

Then on machine A you are told the standby is permitted and the instance is running.

On machine B that standby is permitted and you are running as a standby.

On machine C standby is permitted and is running elsewhere.



Stopping a multi-instance queue manager

- To completely stop a multi-instance queue manager, issue a normal endmqm on the active instance on machine A:

  -
    ```
    endmqm –i QM1
    WebSphere MQ queue manager ‘QM1’ ended.
    ```
  - Both instances end
- To stop just the standby instance, on machine B:

  -
    ```
    endmqm –x QM1
    WebSphere MQ standby queue manager instance ‘QM1’ ended.
    ```
- To switch over the active to the standby, on machine A:

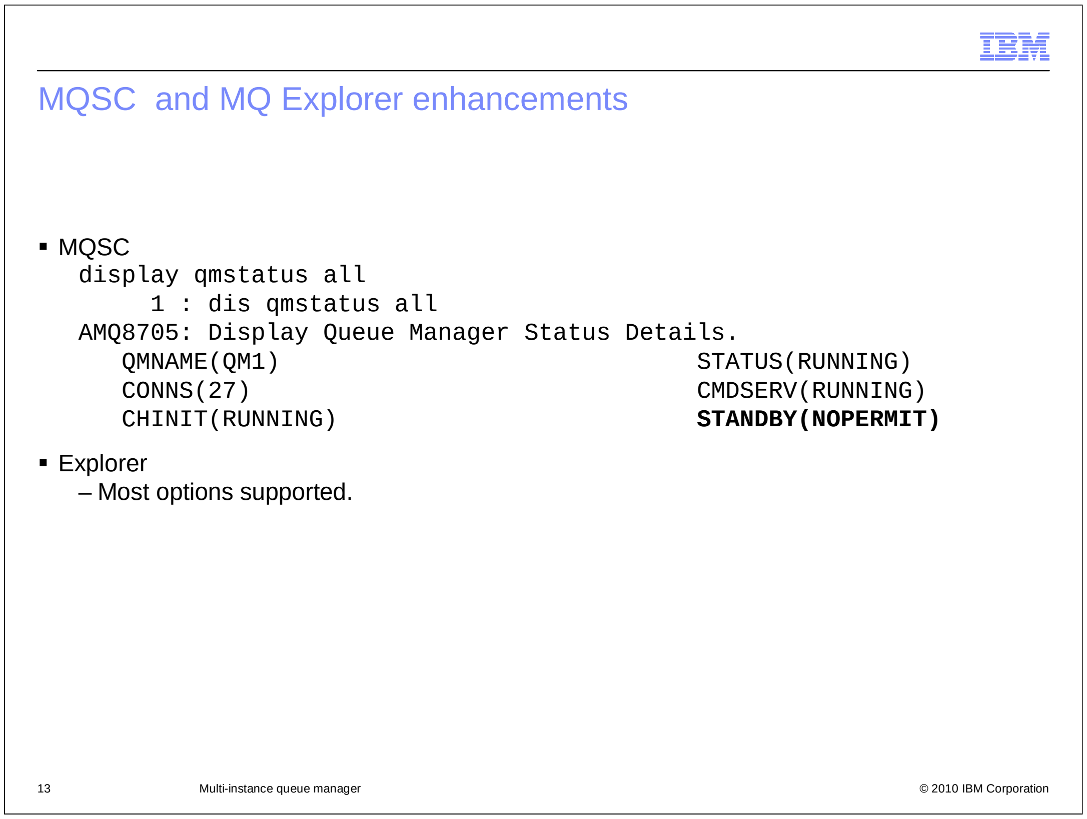
  -
    ```
    endmqm –is QM1
    WebSphere MQ queue manager ‘QM1’ ended, permitting switchover to
    a standby instance
    ```
  - Once the active instance has ended, the standby instance will try to become the active instance.

As well as modifications to the start command to support multi-instance queue managers you also have new options for ending a queue manager.

To stop a multi-instance queue manager you use the “endmqm” command. If the normal endmqm command is issued to an active instance this is taken as a request to fully stop the queue manager. So both the active and the standby instance are ended. This command passed to the standby instance will be ignored.

The new option “-x” can be used to stop only the standby instance, this must be issued on the standby instance.

The new “-s” option can be used to ask an active instance to end only that instance, allowing any standby instance to become active.



MQSC and MQ Explorer enhancements

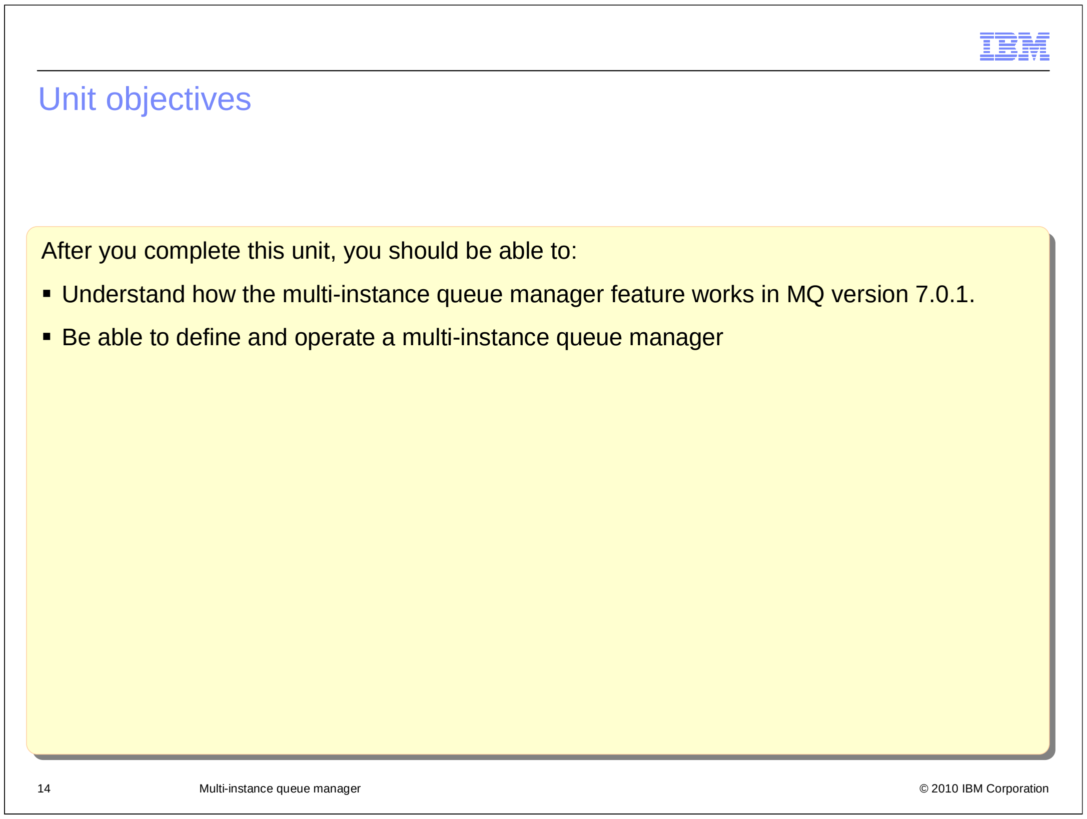
- MQSC

```
display qmstatus all
     1 : dis qmstatus all
AMQ8705: Display Queue Manager Status Details.
   QMNAME(QM1)                             STATUS(RUNNING)
   CONNS(27)                               CMDSERV(RUNNING)
   CHINIT(RUNNING)                         STANDBY(NOPERMIT)
```

- Explorer
  – Most options supported.

In addition to the changes to the above commands the MQSC and explorer interfaces have been extended to support the parameters relevant to multi-instance queue managers.

Similar changes have been made to support the reconnecting client code also.



Unit objectives

After you complete this unit, you should be able to:

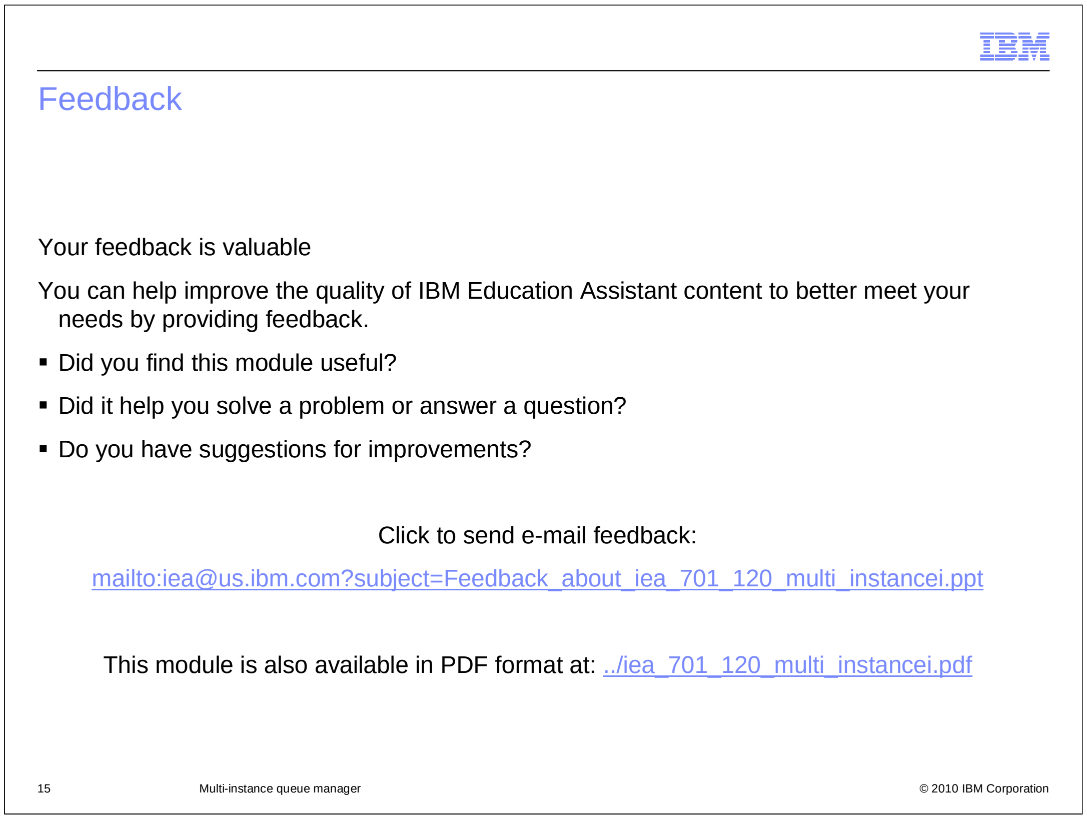
- Understand how the multi-instance queue manager feature works in MQ version 7.0.1.
- Be able to define and operate a multi-instance queue manager

14 Multi-instance queue manager © 2010 IBM Corporation

Now that you have completed this unit you should have some understanding of how MQ 7.0.1 allows multi-instance queue managers to provide basic failover support without an HA coordinator.

How the queue manager data is held in networked storage and how MQ ensures only one instance of the queue manager is active while managing the switch between instances.

This unit does not attempt to cover the full range of syntax and options available, for which you should refer to the product information center.



Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

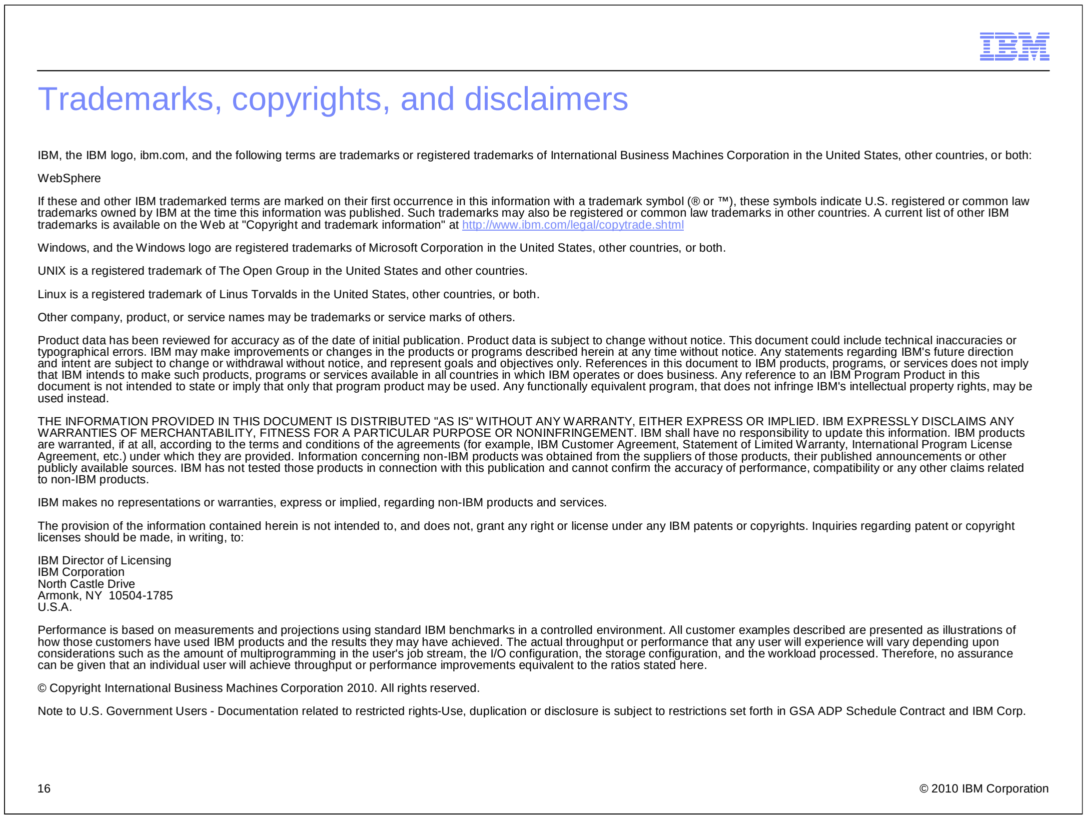
Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_iea_701_120_multi_instancei.ppt

This module is also available in PDF format at: ../iea_701_120_multi_instancei.pdf

15 Multi-instance queue manager © 2010 IBM Corporation

You can help improve the quality of IBM Education Assistant content by providing feedback.



Trademarks, copyrights, and disclaimers