Unit 11. Queue manager clustering

What this unit is about

This unit covers the concepts of queue manager clustering. The discussion covers an overview of components and definitions required for setting up a clustered environment, load balancing through workload management, continuity of service, and administrative commands specific to clustering.

What you should be able to do

After completing this unit, you should be able to:

- Explain the purpose and use of clustering
- Define a cluster and list the components involved
- Describe the difference between a full and a partial repository
- Outline the steps required to set up a basic cluster
- Define the queue manager attributes that are related to the cluster
- List queue and channel definitions
- List the commands used to control clusters
- Describe the administration tasks that must be considered in a clustered environment
- Describe how clusters assist with workload management
- List the attributes that affect the workload balancing algorithm
- Explain how application considerations play a role in workload balancing

How you will check your progress

Accountability:

- Checkpoint
- Machine exercises

References

WebSphere MQ Queue Manager Clusters

Script (MQSC) Command Reference

Unit objectives

After completing this unit, you should be able to:

- Explain the purpose and use of clustering
- Define a cluster and list the components involved
- Describe the difference between a full and a partial repository
- Outline the steps required to set up a basic cluster
- Define the queue manager attributes that are related to the cluster
- List queue and channel definitions
- List the commands used to control clusters
- Describe the administration tasks that must be considered in a clustered environment
- Describe how clusters assist with workload management
- List the attributes that affect the workload balancing algorithm
- Explain how application considerations play a role in workload balancing

Figure 11-1. Unit objectives WM203 / VM2032.0

Notes:

Topics covered

- Cluster overview
- Cluster setup
- Cluster administration
- Workload management

Figure 11-2. Topics covered

Notes:

Cluster overview topic objectives

After completing this topic, you should be able to:
- Explain the purpose and use of clustering
- Define a cluster and list the components involved
- Differentiate between a full and a partial repository
- Highlight the use of clusters in the publish/subscribe function

Figure 11-3. Cluster overview topic objectives WM203 / VM2032.0

Notes:

What a cluster is

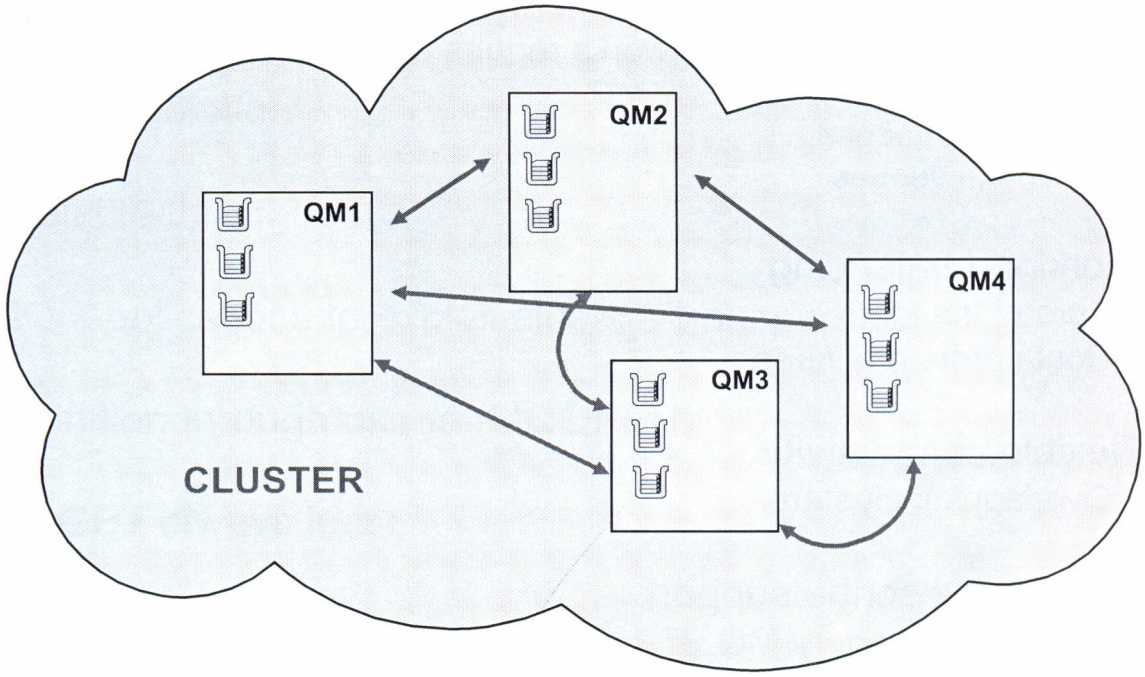


Figure 11-4. What a cluster is

WM203 / VM2032.0

Notes:

A cluster is a collection of queue managers that can be on different platforms, and typically serve a common application.

Every queue manager can make the queues that they host available to every other queue manager in the cluster, without the need for (remote) queue definitions.

Cluster-specific objects remove the need for explicit channel definitions and transmission queues for each destination queue manager.

The queue managers in a cluster often take on the role of a client or a server. The servers host the queues that are available to the members of the cluster, also running applications that process these messages and generate responses. The clients PUT messages to the servers queues and may receive back response messages.

Queue managers in a cluster normally communicate directly with each other, although typically, many of the client systems never need to communicate with other client systems.

The purpose of clustering

- Simplified administration
  - Large WebSphere MQ networks require many object definitions
    - Channels
    - Transmit queues
    - Remote queues
- Workload balancing
  - Spread the load
  - Route around failures
- Flexible connectivity
  - Overlapping clusters
- Publish/Subscribe support
  - New in WebSphere MQ V7

Figure 11-5. The purpose of clustering WM203 / VM2032.0

Notes:

If you do not use clusters, your queue managers are independent and communicate using distributed queuing. If one queue manager needs to send messages to another it must have defined:

- A transmission queue
- A channel to the remote queue manager
- Remote queues

If queue managers are grouped in a cluster, the queue managers can make the queues that they host available to every other queue manager in the cluster. Any queue manager can send a message to any other queue manager in the same cluster without explicit channel definitions, remote-queue definitions, or transmission queues for each destination. Every queue manager in a cluster has a single transmission queue from which it can transmit messages to any other queue manager in the cluster. Each queue manager in a cluster needs to define only:

- One cluster-receiver channel on which to receive messages
- One cluster-sender channel with which it introduces itself and learns about the cluster

Goals of clustering

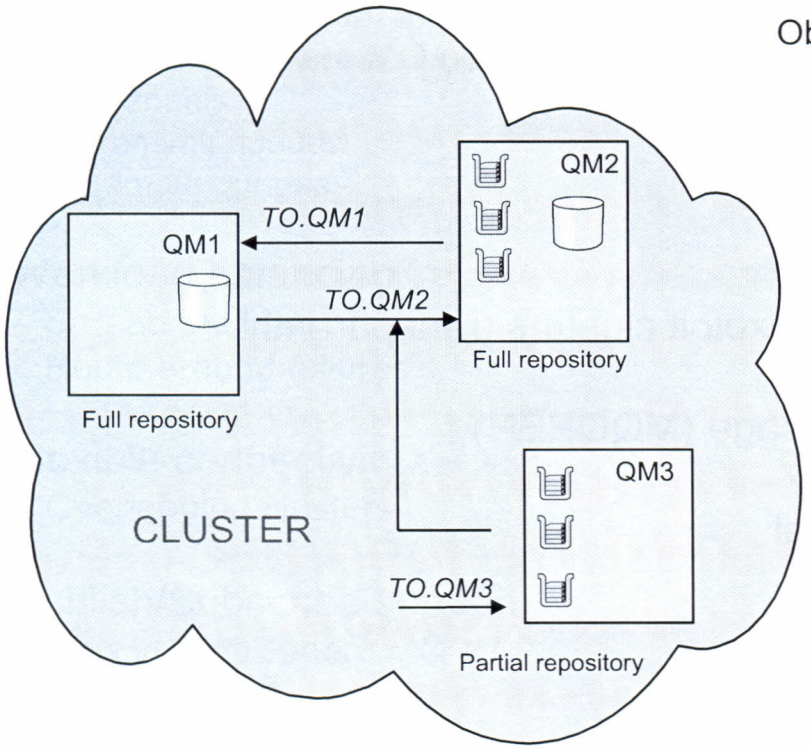
- Multiple queues with a single image
- Failure isolation
- Scalable throughput
- MQI applications to exploit clusters transparently
- Definition through usage (MQOPEN)
- MQGET always local

Figure 11-6. Goals of clustering

WM203 / VM2032.0

Notes:

Overview of cluster components



Objects

- Full repository QMGR
- Partial repository QMGR
- Cluster queues
  - Local and remote
  - Transmission queues
- Cluster channels:
  - Cluster-receiver
  - Cluster-sender

Figure 11-7. Overview of cluster components WM203 / VM2032.0

Notes:

In this cluster there are three queue managers, QM1, QM2, and QM3.

- QM1 and QM2 host repositories of information about the queue managers in the cluster. They are referred to as full repository queue managers. (The repositories are represented in the diagram by the shaded cylinders.)
- QM2 and QM3 host some queues that are accessible to any other queue manager in the cluster. These are called cluster queues. (The cluster queues are represented in the diagram by the shaded queues.) As with distributed queuing, an application uses the MQPUT call to put a message on a cluster queue at any queue manager. An application uses the MQGET call to retrieve messages from a cluster queue on the local queue manager.

Each queue manager has a cluster-receiver channel definition for the receiving end of a channel called TO.*qmgr* on which it can receive messages. A cluster-receiver channel is similar to a receiver channel used in distributed queuing, but in addition to carrying messages this channel can also carry information about the cluster.

Each queue manager also has a cluster-sender channel definition for the sending end of a channel, which connects to the cluster-receiver channel of one of the full repository queue managers. Here, QM1 and QM3 have cluster-sender channels connecting to TO.QM2. QM2 has a cluster-sender channel connecting to TO.QM1. A cluster-sender channel is similar to a sender channel used in distributed queuing, but in addition to carrying messages this channel can also carry information about the cluster. Once both the cluster-receiver end and the cluster-sender end of a channel have been defined, the channel starts automatically.

Cluster channels

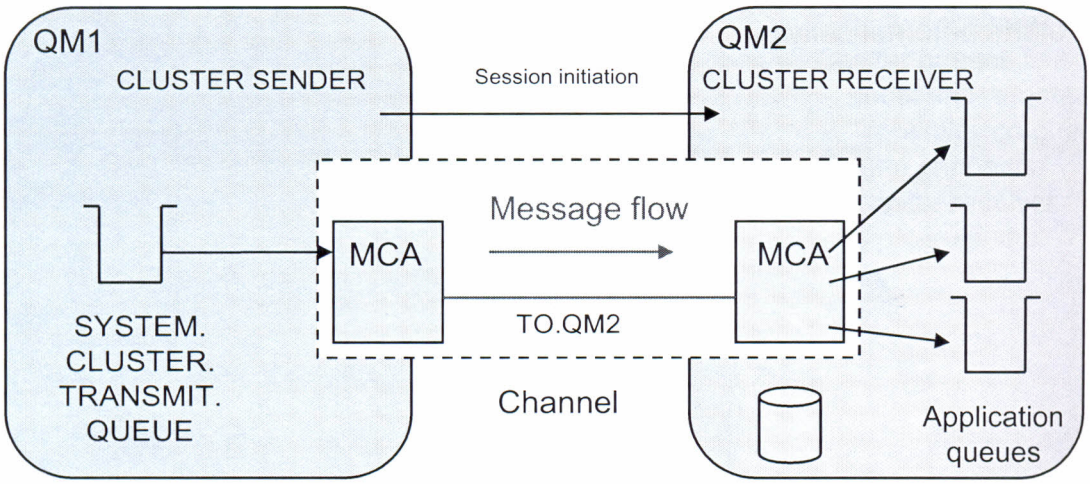


Figure 11-8. Cluster channels WM203 / VM2032.0

Notes:

In a cluster, each queue manager has a cluster-sender channel on which it can send cluster information to one of the full repository queue managers. Queue managers can also send messages to other queue managers on cluster-sender channels.

In a cluster, each queue manager has a cluster-receiver channel on which it can receive messages and information about the cluster.

The SYSTEM.CLUSTER queues

- SYSTEM.CLUSTER.COMMAND.QUEUE
  - Holds inbound administrative messages
  - IPPROCS should always be 1
  - CURDEPTH should be zero or decrementing
  - If not, check repository task and error messages
- SYSTEM.CLUSTER.REPOSITORY.QUEUE
  - Holds hardened view of repository cache
  - CURDEPTH should be greater than zero
  - CURDEPTH varies depending on checkpoints
- SYSTEM.CLUSTER.TRANSMIT.QUEUE
  - Holds outbound administrative messages
  - Holds outbound user messages
  - CorrelId in MQMD added on transmission queue will contain the name of the channel that the message should be sent down

Figure 11-9. The SYSTEM.CLUSTER queues WM203 / VM2032.0

Notes:

To use WebSphere MQ clusters, you need to define the following objects:

- A local queue called the SYSTEM.CLUSTER.COMMAND.QUEUE, which is used to communicate repository changes between queue managers. Messages written to this queue contain updates to the repository data to be applied to the local copy of the repository, or requests for repository data.
- A local queue called SYSTEM.CLUSTER.REPOSITORY.QUEUE, which is used to hold a persistent copy of the repository.
- A local queue called SYSTEM.CLUSTER.TRANSMIT.QUEUE, which is the transmission queue for all destinations in the cluster. For performance reasons, define this queue with an index type of CORRELID (as shown in the sample queue definition).

These queues typically contain large numbers of messages.

These queues are not defined as shared when using shared queue support on z/OS.

Cluster support objects

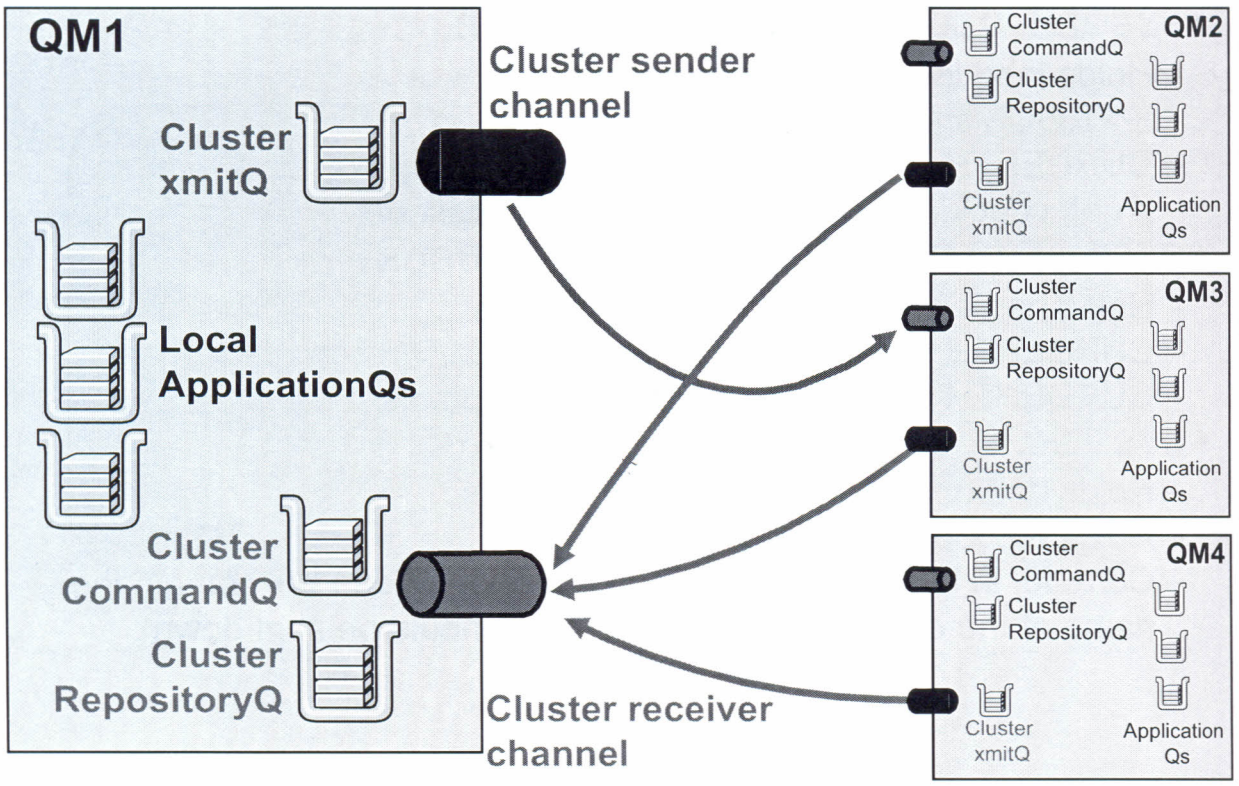


Figure 11-10. Cluster support objects

WM203 / VM2032.0

Notes:

A *cluster repository* (queue) is a collection of information about the queue managers that are members of a cluster, including queue manager names, their channels, the queues they host, and so on. This repository information is exchanged through messages sent to a queue called `SYSTEM.CLUSTER.COMMAND.QUEUE` and *stored* on a queue with the fixed name `SYSTEM.CLUSTER.REPOSITORY.QUEUE`.

Repositories can be full or partial; there is more about this topic on the next visual. Each cluster queue manager must have at least one connection to another queue manager that owns a full repository.

A *cluster-sender channel* is a channel definition of the TYPE(CLUSSDR) on which a cluster queue manager can send messages to another queue manager in the cluster that holds a full repository. This channel is used to notify the repository of any changes of the status of the queue manager, for example the addition or removal of a queue. It is only used for the initial contact with the first full repository queue manager. From this one, the local queue manager learns whatever it needs to know.

Note

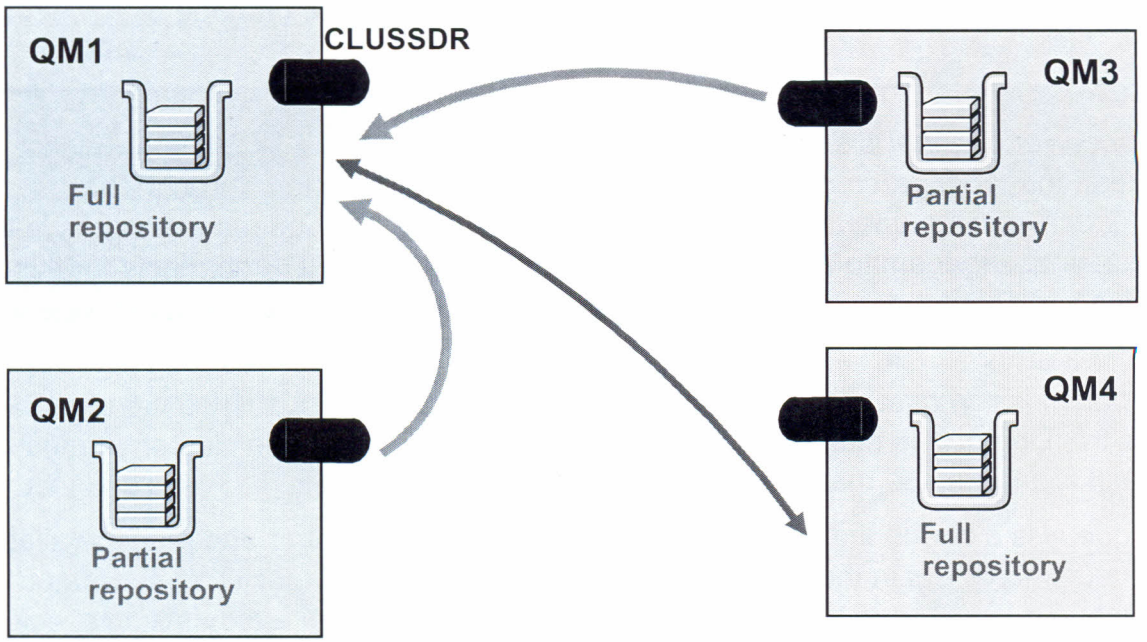
Application messages are sent by auto-defined sender channels that are created during operation based on repository information from other cluster queue managers.

A *cluster-receiver channel* is a channel definition of the TYPE(CLUSRCVR) on which a cluster queue manager can receive messages from within the cluster. Through the definition of this object a queue manager is advertised to the other queue managers in the cluster, thus enabling them to auto-define their appropriate CLUSSDR channels for this queue manager. At least one cluster-receiver channel is required for each cluster queue manager.

A *cluster transmission queue* named `SYSTEM.CLUSTER.TRANSMIT.QUEUE` contains all the messages from the queue manager to any other queue manager in the cluster. It must exist in each cluster queue manager.

A *cluster queue* is a queue that is hosted by a cluster queue manager and made available to other queue managers in the cluster. The local queue is either preexisting or created on the local queue manager and to play a role in the cluster the local queue definition specifies the cluster name of the definition. The other queue managers can see this queue and use it to put messages to it without the use of remote queue definition. The cluster queue can be advertised in more than one cluster.

More about repositories



If QM1 fails, QM2 would connect to QM4 as its full repository system

Figure 11-11. More about repositories

WM203 / VM2032.0

Notes:

- Each cluster queue manager must have a local queue called `SYSTEM.CLUSTER.REPOSITORY.QUEUE` where all cluster-related information is stored.
- At least one (but for availability reasons preferably two or more) cluster queue managers have to hold **full repositories**; that means a complete set of information about every queue manager in the cluster.
- For each cluster queue manager, a **cluster-sender channel** has to be predefined that connects to one of the repository queue managers.
- Repository queue managers (sometimes called *repositories*) must be fully interconnected with each other and positioned in the network to give a high level of availability.
- Normal queue managers build up and maintain a **partial repository** that contains information about those queue managers and queues that are of interest to it. This information may be updated and extended during operation through inquiries of a full repository.

Simplified administration example (1 of 3)

- An example to show the definitions required in a network with four QMGRs, each with two local queues in the configuration:
  - Using distributed queuing (next slide)
  - Using a cluster (following slide)

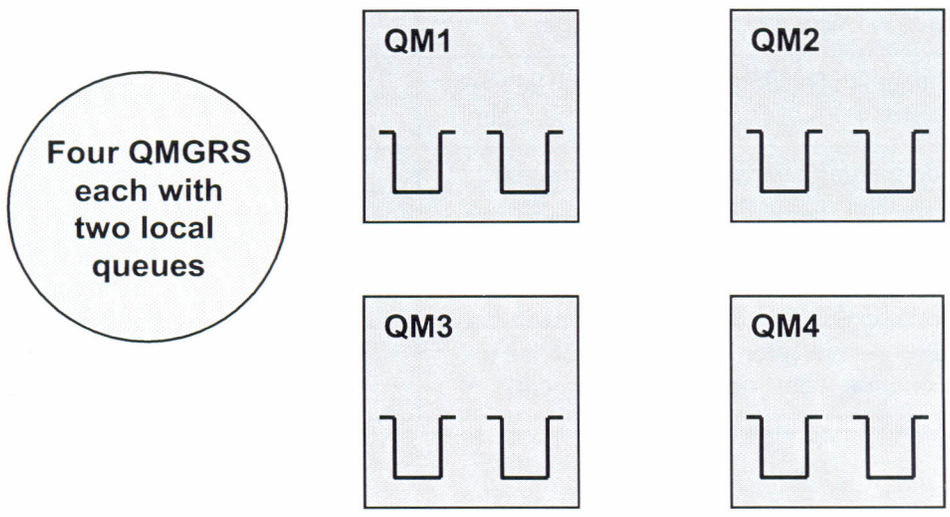


Figure 11-12. Simplified administration example (1 of 3) WM203 / VM2032.0

Notes:

Using clusters leads to easier administration of a network. This slide shows four queue managers each with two queues. Consider how many definitions are needed to connect these queue managers using distributed queuing. You see how many definitions are needed to set up the same network as a cluster.

Simplified administration example (2 of 3)

- Definitions required using distributed queuing

Description	Number per QMGR	Total number
A sender-channel definition for a channel on which to send messages to every other queue manager	3	12
A receiver-channel definition for a channel on which to receive messages from every other queue manager	3	12
A transmission-queue definition for a transmission queue to every other queue manager	3	12
A local-queue definition for each local queue	2	8
A remote-queue definition for each remote queue to which this queue manager wants to put messages	6	24

Figure 11-13. Simplified administration example (2 of 3) WM203 / VM2032.0

Notes:

While you might reduce this number of definitions by, for example, using generic receiver-channel definitions, the maximum number of definitions can be as many as 17 on each queue manager, which is a total of 68 for this network.

Simplified administration example (3 of 3)

When using clusters, you need:

- Just one CLUSSDR and one CLUSRCVR definition at each queue manager
- No separately defined transmission queues
- No remote-queue definitions

Description	Number per QMGR	Total number
A cluster-sender channel definition for a channel on which to send messages to a repository queue manager	1	4
A cluster-receiver channel definition for a channel on which to receive messages from other queue managers in the cluster	1	4
A local-queue definition for each local queue	2	8

Figure 11-14. Simplified administration example (3 of 3) WM203 / VM2032.0

Notes:

To set up this cluster of queue managers (with two full repositories), you would need four definitions on each queue manager — a total of 16 definitions all together. You would also need to alter the queue-manager definitions for two of the queue managers, to make them full repository queue managers for the cluster.

The CLUSSDR and CLUSRCVR channel definitions need be made only once. When the cluster is in place you can add or remove queue managers (other than the repository queue managers) without any disruption to the other queue managers. This process significantly reduces the number of definitions required to set up a network containing many queue managers.

With fewer definitions to make there is less risk of error:

- Object names always match, for example the channel name in a sender-receiver pair.
- The transmission queue name specified in a channel definition always matches the correct transmission queue definition or the transmission queue name specified in a remote queue definition.

- A QREMOTE definition always points to the correct queue at the remote queue manager.

Once a cluster is set up, cluster queues can be moved from one queue manager to another within the cluster without having to perform any system management work on any other queue manager. There is no chance of forgetting to delete or modify channel, remote-queue, or transmission-queue definitions. You can add new queue managers to a cluster without any disruption to the existing network.

Publish/subscribe clusters (1 of 2)

- For distributed publish/subscribe
  - New in WebSphere MQ V7
- Based on clustered topic objects
  - Hosted on one or more queue managers in the cluster
- Based on clustering for object auto-definition
  - All to all queue manager connectivity
  - Channel auto-definition on first cluster topic definition
  - Topics can be advertised to the cluster using the CLUSTER() keyword

```
DEFINE TOPIC(SPORTS) TOPICSTR(/global/sports) CLUSTER(DEMO)
```

- How large?
  - 100's of queue managers

Figure 11-15. Publish/subscribe clusters (1 of 2) WM203 / VM2032.0

Notes:

Publish/subscribe clusters (2 of 2)

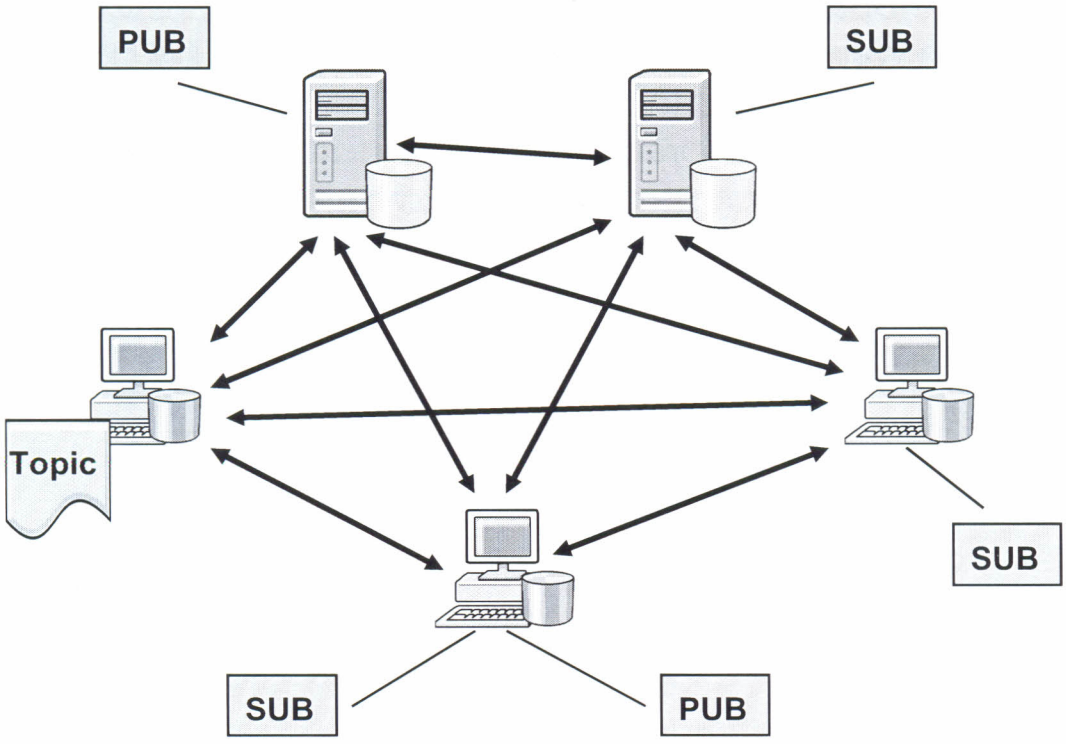


Figure 11-16. Publish/subscribe clusters (2 of 2) WM203 / VM2032.0

Notes:

Cluster overview topic summary

Having completed this topic, you should be able to:
- Explain the purpose and use of clustering
- Define a cluster and list the components involved
- Differentiate between a full and a partial repository
- Highlight the use of clusters in the publish/subscribe function

Figure 11-17. Cluster overview topic summary WM203 / VM2032.0

Notes:

Cluster setup topic objectives

After completing this topic, you should be able to:
- Outline the steps required to setup a basic cluster
- Define the cluster related queue manager attributes
- List the necessary queue and channel definitions

Figure 11-18. Cluster setup topic objectives

WM203 / VM2032.0

Notes:

Steps to set up a cluster

1. Decide how many queue managers to define (organization of the cluster and naming conventions)
2. Establish which queue managers are to hold full repositories
3. Alter the full repository QMGR definitions to add the repository information
4. Define the CLUSRCVR channels
5. Define the CLUSSDR channels
6. Define the cluster queues
7. Verify and test the cluster

Figure 11-19. Steps to set up a cluster WM203 / VM2032.0

Notes:

The figure lists the basic steps for setting up a cluster.

Considerations for a full repository

- Full repositories (FRs) should be highly available
  - Avoid single point of failure
  - Recommended to have two
  - Locate on highly available machines
- FRs must be fully interconnected
  - Use manually defined cluster sender channels
- If at least one FR is not available or not fully connected
  - Cluster definition changes using FRs will not flow
  - User messages between partial repositories over existing channels will flow

Figure 11-20. Considerations for a full repository WM203 / VM2032.0

Notes:

In each cluster you must select at least one, preferably two, or possibly more of the queue managers to hold full repositories. A cluster can work adequately with only one full repository but using two improves availability. You interconnect the full repository queue managers by defining cluster-sender channels between them.

When a queue manager sends out information about itself or requests information about another queue manager, the information or request is sent to two full repositories. A full repository named on a CLUSSDR definition handles the request whenever possible, but if the chosen full repository is not available another full repository is used. When the first full repository becomes available again it collects the latest new and changed information from the others so that they keep in step.

Add repository definition to QMGR

- On each queue manager that is to hold a full repository, you need to alter the queue-manager definition, using the ALTER QMGR command and specifying the REPOS attribute:
- Use `runmqsc` command:

```
     1 : ALTER QMGR REPOS(MYCLUSTER)
AMQ8005: Websphere MQ queue manager changed.
```


Figure 11-21. Add repository definition to QMGR WM203 / VM2032.0

Notes:

If you just runmqsc and enter the ALTER QMGR command, the local queue manager is changed.

Cluster related queue manager attributes

REPOS	Cluster name
REPOSNL	Name list name
CLWLDATA	Cluster workload exit data (32 character maximum string)
CLWLEXIT	Cluster workload exit name
CLWLLEN	Maximum number of bytes of message data passed to cluster workload exit

Figure 11-22. Cluster related queue manager attributes WM203 / VM2032.0

Notes:

The list shows the attributes of a queue manager that can be altered to change some cluster management options. You can:

- In the *repository management* section, specify the name of a single cluster or of a cluster namelist for which this queue manager is to provide repository manager services.
- Determine the name of an optional user exit program that customizes auto-definition of cluster sender channels.
- Determine the name of the cluster workload exit program that can be called whenever a message is put to a cluster queue.
- Specify a 32-character data string and the maximum bytes of message data that can be passed to the workload exit.

Define CLUSRCVR channels

- One cluster receiver channel for each QMGR
  - To receive messages
  - Use `CHLTYPE(CLUSRCVR)` keyword
- Defines QMGR connection name
  - Stored in the repository for access from other QMGRs
- `CLUSTER` keyword
  - Shows the QGMR is available in the cluster
- Example using transport protocol TCP/IP:

```
DEFINE CHANNEL(TO.MYQMGR1) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) +
   CONNAME(MYQMGR1.CHSTORE.COM) CLUSTER(MYCLUSTER) +
   DESCR('TCP Cluster-receiver channel for queue manager MYQMGR1')
```


Figure 11-23. Define CLUSRCVR channels WM203 / VM2032.0

Notes:

In this example, the channel name is TO.MYQMGR1, and the connection name (CONNAME) is the network address of the computer the queue manager resides on, which is MYQMGR1.CHSTORE.COM. The network address can be entered as an alphanumeric DNS host name, or an IP address.

The port number is not specified, so the default port (1414) is used.

Define CLUSSDR channel

- Need one cluster sender channel for each QMGR in the cluster
  - Sends messages to the cluster full repository QMGRs
- Channel names given on the CLUSSDR definitions must match those on the corresponding CLUSRCVR definitions
- Example using transport protocol TCP/IP

```
DEFINE CHANNEL(TO.MYQMGR2) CHLTYPE(CLUSSDR) TRPTYPE(TCP) +
  CONNAME(MYQMGR2.CHSTORE.COM) CLUSTER(MYCLUSTER) +
  DESCR('TCP Cluster-sender channel from MYQMGR1 +
        to repository at MYQMGR2')
```


Figure 11-24. Define CLUSSDR channel WM203 / VM2032.0

Notes: