10.0

Planning for IBM MQ

Note

Before using this information and the product it supports, read the information in “Notices” on page 203.

This edition applies to version 9 release 4 of IBM® MQ and to all subsequent releases and modifications until otherwise indicated in new editions.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 2007, 2026.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Planning.....5
IBM MQ release types: planning considerations..... 6
IBM MQ and IBM MQ Appliance on premises considerations for GDPR readiness..... 8
Architectures based on a single queue manager.....18
Architectures based on multiple queue managers.....19
Planning your distributed queues and clusters.....19
Planning your distributed publish/subscribe network..... 68
Planning your storage and performance requirements on Multiplatforms.....106
Disk space requirements on Multiplatforms..... 107
Planning file system support on Multiplatforms..... 109
Planning file system support for MFT on Multiplatforms.....135
Choosing circular or linear logging on Multiplatforms..... 135
Shared memory on AIX.....136
IBM MQ and UNIX System V IPC resources.....137
IBM MQ and UNIX Process Priority..... 137
Planning your IBM MQ environment on z/OS.....137
Planning for your queue manager.....138
Planning your channel initiator.....166
Planning your queue sharing group (QSG)..... 171
Planning for backup and recovery..... 184
Planning your z/OS UNIX environment.....193
Planning for Advanced Message Security on z/OS.....193
Planning for Managed File Transfer on z/OS..... 194
Planning to use the IBM MQ Console and REST API on z/OS 200

Notices.....203
Programming interface information.....204
Trademarks..... 204

Planning an IBM MQ architecture

When planning your IBM MQ environment, consider the support that IBM MQ provides for single and multiple queue manager architectures, and for point-to-point and publish/subscribe messaging styles. Also plan your resource requirements, and your use of logging and backup facilities.

About this task

Before you plan your IBM MQ architecture, familiarize yourself with the basic IBM MQ concepts. See IBM MQ Technical overview.

IBM MQ architectures range from simple architectures using a single queue manager, to more complex networks of interconnected queue managers. Multiple queue managers are connected together using distributed queuing techniques. For more information about planning single queue manager and multiple queue manager architectures, see the following topics:

- “Architectures based on a single queue manager” on page 18
- “Architectures based on multiple queue managers” on page 19
  - “Planning your distributed queues and clusters” on page 19
  - “Planning your distributed publish/subscribe network” on page 68

z/OS On IBM MQ for z/OS® you can use shared queues and queue sharing groups to enable you to implement workload balancing, and your IBM MQ applications to be scalable and highly available. For information about shared queues and queue sharing groups, see Shared queues and queue sharing groups.

IBM MQ provides two different release models:

- The Long Term Support (LTS) release is most suitable for systems requiring a long term deployment and maximum stability.
- The Continuous Delivery (CD) release is intended for systems which need to rapidly exploit the latest functional enhancements for IBM MQ.

Both release types are installed in the same way, but there are considerations relating to support and migration which you need to understand. For more information, see IBM MQ release types and versioning.

For information about planning for multiple installations, storage and performance requirements, and use of clients, see the other subtopics.

Related concepts

IBM MQ release types and versioning

“Planning your IBM MQ environment on z/OS” on page 137
When planning your IBM MQ environment, you must consider the resource requirements for data sets, page sets, Db2, Coupling Facilities, and the need for logging, and backup facilities. Use this topic to plan the environment where IBM MQ runs.

Availability, recovery and restart

Related tasks

Checking requirements

Making sure that messages are not lost (logging)

© Copyright IBM Corp. 2007, 2026

IBM MQ release types: planning considerations

The two main release types for IBM MQ are Long Term Support (LTS) and a Continuous Delivery (CD). For each supported platform, the release type you choose affects ordering, installation, maintenance and migration.

For detailed information about the release types, see IBM MQ release types and versioning.

Considerations for IBM MQ for Multiplatforms

Ordering

Within Passport Advantage® there are two separate eAssemblies for IBM MQ 10.0. One contains installation images for IBM MQ 10.0.0 Long Term Support release, and the other contains installation images for IBM MQ 10.0.x Continuous Delivery release. Download installation images from the eAssembly according to your choice of release.

All IBM MQ versions, and for IBM MQ 10.0 both the LTS releases and CD releases, belong to the same Product Id.

Entitlement to use IBM MQ extends across the entire product (PID), subject to the constraints of licensed components and pricing metrics. This means that you can choose freely between LTS release and CD release installation images for IBM MQ 10.0.

Installation

After you download an installation image from Passport Advantage, you should select for installation only the components for which you have purchased entitlement. See IBM MQ license information for further information about which installable components are included for each chargeable component.

You can install IBM MQ 10.0.0 LTS release and IBM MQ 10.0.x CD release on the same operating system image. If you do this, the components appear as separate installations, as supported by IBM MQ multi-version support. Each version has distinct sets of queue managers associated with that version.

Each new CD release is provided as an installation image. The new CD release can be installed alongside an existing release, or, an earlier CD release can be updated in place by the installer to the new release.

CD releases contain functional enhancements as well as the latest set of defect fixes and security updates. Each CD release is cumulative, and wholly replaces all prior ones for that version of IBM MQ. So you can skip a specific CD release if it does not contain any function that is relevant for your enterprise.

Maintenance

The LTS release is serviced by the application of fix packs, which provide defect fixes, and cumulative security updates (CSUs), which provide security patches. The fix packs and CSUs are made available periodically and are cumulative.

For CD, CSUs are produced for the latest CD release only, which might be on a subsequent version.

You might occasionally be directed by IBM Support to apply an interim fix. Interim fixes are also known as emergency or test fixes, and are used to apply urgent updates that cannot wait for the next maintenance delivery.

Migration between LTS release and CD release

There are constraints and limitations but, generally, a single queue manager can be migrated from using LTS release code to CD release code, or from using CD release code to LTS release code, provided that the target release is higher than that in use prior to the migration.

Two approaches are possible:

- Install the new release of code in place so that an existing installation of IBM MQ is updated. Any queue managers associated with the installation use the new release of code when started.
- Install the new release of code as a new installation, then move individual queue manager instances to the new installation using the setmqm command.

When a queue manager starts running a CD release of code, the queue manager command level is updated to indicate the new release level. This means any new functions provided in the release are enabled, and you can no longer restart the queue manager using a code release with a lower VRM number.

Considerations for IBM MQ for z/OS

z/OS

Ordering

When ordering IBM MQ for z/OS 10.0, two separate features are offered on ShopZ. The features correspond to the LTS release and the CD release. Both features are applicable to the same product ID (PID). It is the product ID that is licensed, so where one feature is licensed, there is entitlement to use the alternative feature if required. When ordering, you select the feature corresponding with either the LTS release or CD release.

If you are selecting products for inclusion in a ServerPac, you cannot choose both the LTS release and CD release in the same ServerPac order, because the products cannot be installed by SMP/E in the same target zone.

Installation

The LTS and CD releases are provided in separate sets of FMIDs. Note that these FMIDs cannot be installed in the same SMP/E target zone. If you need both the LTS and CD releases:

- Install the LTS release and CD release in separate target zones.
- Maintain separate target and distribution libraries for the two releases.

The command level of a queue manager is the four-digit VVRM level. An IBM MQ program can call `MQINQ`, passing the `MQIA_COMMAND_LEVEL` selector, to get the command level of the queue manager it is connected to.

Because the releases use different FMIDs, you cannot update a CD release with maintenance for an LTS release or the other way round. Similarly, there is no way to switch a version of the product code from an LTS release to a CD release or the other way round. However, you can switch a queue manager between the release models. See Migration between LTS release and CD release.

The IBM MQ 10.x.x CD releases use a different set of FMIDs to the IBM MQ 9.x.x CD releases. Therefore moving from IBM MQ 9.x.x CD to IBM MQ 10.x.x CD will require a new SMP/E install. All of the IBM MQ 10.x.x CD releases will use the same set of FMIDs, meaning you can upgrade IBM MQ 10.x.x CD releases by applying PTFs to a single SMP/E install even when crossing a major version boundary. For example, in future, you can go from IBM MQ for z/OS 10.0.0 CD, to IBM MQ for z/OS 10.0.1 CD, to IBM MQ for z/OS 10.1.0 CD just by applying PTFs.

Maintenance

IBM MQ for z/OS uses PTFs for Maintenance.

LTS PTFs are specific to a particular set of libraries corresponding to specific release level. For UNIX System Services features (that is, JMS and WEB UI, Connector Pack, and Managed File Transfer) the z/OS PTFs are aligned directly with the Multiplatforms fix packs and cumulative security updates (CSUs). These fixes are cumulative, and are made available at the same time as the equivalent Multiplatforms fix pack or CSU.

CD CD CSUs are not usually made available between CD releases, but are included in the next IBM MQ for z/OS CD release. You can also contact support to request a ++USERMOD.

Other fixes on IBM MQ for z/OS are distinct fixes on particular parts. These fixes resolve specific issues, are not cumulative, and are made available as and when they are produced.

Migration between LTS release and CD release

There are constraints and limitations, but generally a single queue manager can be migrated from using LTS release code to CD release code or from using CD release code to LTS release code provided that the target release is higher than that in use prior to the migration.

From IBM MQ for z/OS 9.2.0, you can migrate back and forth between CD and LTS releases with the same VRM as many times as needed, and without impact on the ability to backwards migrate. For example, a queue manager can be started up at IBM MQ for z/OS 9.3.0 LTS, then shut down and started up at IBM MQ for z/OS 9.3.0 CD, then shut down and started up at IBM MQ for z/OS 9.3.0 LTS.

IBM MQ for z/OS has traditionally provided a fallback capability (backward migration) so that after a period of running following a migration you can fall back to the prior release. This capability is retained for LTS releases, and those CD releases with a modifier of `0` such as 9.3.0 CD, but is not possible when the source or target of a migration is a CD release with a non-zero modifier number, for example, 9.2.5 or 9.3.1.

The following are valid migration scenarios, and illustrate how this principle works:

Source release	Destination release	Notes
9.2.0 LTS	10.0.0 LTS or 10.0.0 CD	Backwards migration not supported as 9.2.0 LTS is out of standard support.
9.3.0 LTS	10.0.0 LTS or 10.0.0 CD	Backwards migration supported.
9.4.0 LTS	10.0.0 LTS or 10.0.0 CD	Backwards migration supported.
9.4.5 CD	10.0.0 LTS or 10.0.0 CD	Backwards migration not supported as source release is CD with a non-zero modifier.
10.0.0 LTS or 10.0.0 CD	10.0.1 CD	Backwards migration not supported as destination release is CD with a non-zero modifier. `Write to operator with reply` CSQY041D is issued to confirm migration.

Related concepts

Migration paths

Related tasks

z/OS Applying and removing maintenance on z/OS

Related information

Downloading IBM MQ 10.0

IBM MQ and IBM MQ Appliance on premises considerations for GDPR readiness

For PID(s):

Distributed

- IBM MQ/IBM MQ Advanced - 5724-H72
- IBM MQ for HPE NonStop - 5724-A39

z/OS

- IBM MQ for z/OS - 5656-MQA
- IBM MQ for z/OS Value Unit Edition - 5656-VUA
- IBM MQ Advanced for z/OS - 5656-ADA
- IBM MQ Advanced for z/OS Value Unit Edition - 5656-AVA

IBM MQ Appliance

- IBM MQ Appliance M2003 - 5900-ALJ
- IBM MQ Appliance M2002 - 5737-H47

Notice:

This document is intended to help you in your preparations for GDPR readiness. It provides information about features of IBM MQ that you can configure, and aspects of the product's use, that you should consider to help your organization with GDPR readiness. This information is not an exhaustive list, due to the many ways that clients can choose and configure features, and the large variety of ways that the product can be used in itself and with third-party applications and systems.

Clients are responsible for ensuring their own compliance with various laws and regulations, including the European Union General Data Protection Regulation. Clients are solely responsible for obtaining advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulations that may affect the clients' business and any actions the clients may need to take to comply with such laws and regulations.

The products, services, and other capabilities described herein are not suitable for all client situations and may have restricted availability. IBM does not provide legal, accounting, or auditing advice or represent or warrant that its services or products will ensure that clients are in compliance with any law or regulation.

Table of Contents

1. GDPR
2. Product Configuration for GDPR
3. Data Life Cycle
4. Data Collection
5. Data Storage
6. Data Access
7. Data Processing
8. Data Deletion
9. Data Monitoring
10. Capability for Restricting Use of Personal Data
11. File handling

GDPR

General Data Protection Regulation (GDPR) has been adopted by the European Union ("EU") and applies from May 25, 2018.

Why is GDPR important?

GDPR establishes a stronger data protection regulatory framework for processing of personal data of individuals. GDPR brings:

- New and enhanced rights for individuals
- Widened definition of personal data
- New obligations for processors
- Potential for significant financial penalties for non-compliance
- Compulsory data breach notification

Read more about GDPR:

- EU GDPR Information Portal

- ibm.com/GDPR website

Product Configuration - considerations for GDPR Readiness

The following sections provide considerations for configuring IBM MQ to help your organization with GDPR readiness.

Data Life Cycle

IBM MQ is a transactional message oriented middleware product that enables applications to asynchronously exchange application provided data. IBM MQ supports a range of messaging APIs, protocols and bridges for the purpose of connecting applications. As such, IBM MQ may be used to exchange many forms of data, some of which could potentially be subject to GDPR. There are several third-party products with which IBM MQ might exchange data. Some of these are IBM-owned, but many others are provided by other technology suppliers. The Software Product Compatibility Reports website provides lists of the associated software. For considerations regarding the GDPR readiness of a third-party product, you should consult that product's documentation. IBM MQ administrators control the way in which IBM MQ interacts with data passing through it, by the definition of queues, topics and subscriptions.

What types of data flow through IBM MQ?

As IBM MQ provides asynchronous messaging service for application data, there is no one definitive answer to this question because use cases vary through application deployment. Application message data is persisted in queue files (page sets or the Coupling Facility on z/OS), logs and archives and the message may itself contain data that is governed by GDPR. Application provided message data may also be included in files collected for problem determination purposes such as error logs, trace files and FFSTs. On z/OS application provided message data may also be included in address space or Coupling Facility dumps.

The following are some typical examples of personal data that may be exchanged using IBM MQ:

- Employees of the customer (for example; IBM MQ might be used to connect the customer's payroll or HR systems)
- The customer's own clients' personal data (for example; IBM MQ might be used by a customer to exchange data between applications that relates to their clients, such as taking sales leads and storing data inside their CRM system).
- The customer's own clients' sensitive personal data (for example; IBM MQ might be used within industry contexts that require personal data to be exchanged, such as HL7-based healthcare records when integrating clinical applications).

In addition to application provided message data, IBM MQ processes the following types of data:

- Authentication Credentials (such as username and passwords, API keys, etc.)
- Technically Identifiable Personal Information (such as device IDs, usage based identifiers, IP address, etc. - when linked to an individual)

Personal data used for online contact with IBM

IBM MQ clients can submit online comments/feedback/requests to contact IBM about IBM MQ subjects in a variety of ways, primarily:

- Public comments area on pages in the IBM MQ area on IBM Developer
- Public comments area on pages of IBM MQ product information in IBM Documentation
- Public comments in the IBM Support Forums
- Public comments in IBM Integration Ideas

Typically, only the client name and email address are used, to enable personal replies for the subject of the contact, and the use of personal data conforms to the IBM Online Privacy Statement.

Data Collection

IBM MQ can be used to collect personal data. When assessing your use of IBM MQ and your needs to meet with the demands of GDPR, you should consider the types of personal data which in your circumstances are passing through IBM MQ. You may wish to consider aspects such as:

- How does data arrive into your queue managers? (Across which protocols? Is the data encrypted? Is the data signed?)
- How is data sent out from your queue managers? (Across which protocols? Is the data encrypted? Is the data signed?)
- How is data stored as it passes through a queue manager? (Any messaging application has the potential to write message data to stateful media, even if a message is non-persistent. Are you aware of how messaging features could potentially expose aspects of the application message data passing through the product?)
- How are credentials collected and stored where needed by IBM MQ to access third-party applications?

IBM MQ may need to communicate with other systems and services which require authentication, for example LDAP. Where needed, authentication data (userids, passwords) is configured and stored by IBM MQ for its use in such communications. Wherever possible, you should avoid using personal credentials for IBM MQ authentication. Consider the protection of the storage used for authentication data. (See Data Storage below.)

Data Storage

When message data travels through queue managers, IBM MQ will persist (perhaps multiple copies of) that data directly to stateful media. IBM MQ users may want to consider securing message data whilst it is at rest.

The following items highlight areas where IBM MQ persists application provided data, which users may wish to consider when ensuring compliance with GDPR.

- Application Message Queues:

  IBM MQ provides message queues to allow asynchronous data exchange between applications. Non-persistent and persistent messages stored on a queue are written to stateful media.

- File Transfer Agent Queues:

  IBM MQ Managed File Transfer utilizes message queues to co-ordinate the reliable transfer of file data, files that contain personal data and records of transfers are stored on these queues.

- Transmission Queues:

  To transfer messages reliably between queue managers, messages are stored temporarily on transmission queues.

- Dead-Letter Queues:

  There are some circumstances in which messages cannot be put to a destination queue and are stored on a dead-letter queue, if such a queue is configured on the queue manager.

- Backout Queues:

  JMS and XMS messaging interfaces provide a capability that allows poison messages to be moved to a backout queue after a number of backouts have occurred to allow other valid messages to be processed.

- AMS Error Queue:

  IBM MQ Advanced Message Security will move messages that don't comply with a security policy to the SYSTEM.PROTECTION.ERROR.QUEUE error queue in a similar way to dead-letter queuing.

- Retained Publications:

  IBM MQ provides a retained publication feature to allow subscribing applications to recall a previous publication.

- Deferred Delivery:

  IBM MQ supports the JMS 2.0 and Jakarta Messaging 3.0 delivery delay feature that enables messages to be delivered to their destination at a future time. Messages that have not yet been delivered are stored on the SYSTEM.DDELAY.LOCAL.QUEUE queue.

Read more:

- Logging: Making sure that messages are not lost
- MFT Agent queue settings
- Using the dead-letter queue
- Handling poison messages in IBM MQ classes for JMS
- AMS error handling
- Retained publications
- JMS 2.0 delivery delay

The following items highlight areas where IBM MQ may indirectly persist application provided data which users may also wish to consider when ensuring compliance with GDPR.

- Trace route messaging:

  IBM MQ provides trace route capabilities, which record the route a message takes between applications. The event messages generated may include technically identifiable personal information such as IP addresses.
- Application activity trace:

  IBM MQ provides application activity trace, which record the messaging API activities of applications and channels, application activity trace can record the contents of application provided message data to event messages.
- Service trace:

  IBM MQ provides service trace features, which records the internal code paths through which message data flows. As part of these features, IBM MQ can record the contents of application provided message data to trace files stored on disk.
- Queue manager events:

  IBM MQ can generate event messages that could include personal data, such as authority, command and configuration events.

Read more:

- Trace-route messaging
- Using trace
- Event monitoring
- Queue manager events

To protect access to copies of the application provided message data consider the following actions:

- Restrict privileged user access to IBM MQ data in the filesystem, for example restricting user membership of the 'mqm' group on UNIX and Linux® platforms.
- Restrict application access to IBM MQ data via dedicated queues and access control. Where appropriate avoid unnecessary sharing of resources such as queues between applications and provide granular access control to queue and topic resources.
- Restrict access to replicated copies of IBM MQ data in high availability (HA) or disaster recovery (DR) configurations, and secure the connections used for replication.
- Use IBM MQ Advanced Message Security to provide end-to-end signing and/or encryption of message data.
- Use file- or volume-level encryption to protect directories or filesystems that might contain IBM MQ data, trace or logs.

- After uploading service trace to IBM, you can delete your service trace files and FFST data if you are concerned about the contents potentially containing personal data.

Read more:

- Privileged users
- Planning file system support on Multiplatforms
- Confidentiality for data at rest on IBM MQ for z/OS with data set encryption
- Planning storage for the IBM MQ Operator - Encryption
- File system encryption on the IBM MQ Appliance

An IBM MQ administrator may configure a queue manager with credentials (username and password, API keys, etc.) for third-party services such as LDAP. This data is generally stored in the queue manager data directory protected through file system permissions.

When an IBM MQ queue manager is created, the data directory is set up with group-based access control such that IBM MQ can read the configuration files and use the credentials to connect to these systems. IBM MQ administrators are considered privileged users and are members of this group so have read access to the files. Some files are obfuscated but they are not encrypted. For this reason, to fully protect access to credentials, you should consider the following actions:

- Restrict privileged user access to IBM MQ data, for example restricting membership of the 'mqm' group on UNIX and Linux platforms.
- Use file- or volume-level encryption to protect the contents of the queue manager data directory.
- Encrypt backups of the production configuration directory and store them with appropriate access controls.
- Consider providing audit trails for authentication failure, access control and configuration changes with security, command and configuration events.

Read more:

- Securing IBM MQ

Data Access

IBM MQ queue manager data can be accessed through the following product interfaces, some of which are designed for access through a remote connection, and others for access through a local connection.

- IBM MQ Console [Only Remote]
- IBM MQ Administrative REST API [Only Remote]
- IBM MQ Messaging REST API [Only Remote]
- MQI [Local and Remote]
- JMS [Local and Remote]
- XMS [Local and Remote]
- IBM MQ Telemetry (MQTT) [Only Remote]
- IBM MQ Light (AMQP) [Only Remote]
- IBM MQ IMS bridge [Only Local]
- IBM MQ CICS bridge [Only Local]
- IBM MQ MFT Protocol bridges [Only Remote]
- IBM MQ Connect:Direct bridges [Only Remote]
- IBM MQ MQAI [Local and Remote]
- IBM MQ PCF commands [Local and Remote]
- IBM MQ MQSC commands [Local and Remote]
- IBM MQ Explorer [Local and Remote]

- IBM MQ User Exits [Only Local]
- IBM MQ Internet Pass-Thru [Only Remote]
- Red Hat® OpenShift® Monitoring (Prometheus) metrics (the metrics are numerical data about queue manager statistics)
- IBM MQ Appliance Serial Console [Only Local]
- IBM MQ Appliance SSH [Only Remote]
- IBM MQ Appliance REST API [Only Remote]
- IBM MQ Appliance Web UI [Only Remote]
- IBM MQ Kafka Connectors (Kafka Connect) [Local and Remote]
- IBM MQ Agent [Only Remote]

The interfaces are designed to allow users to make changes to an IBM MQ queue manager and messages stored on it. Administration and messaging operations are secured such that there are three stages involved when a request is made;

- Authentication
- Role mapping
- Authorization

Authentication:

If the message or administrative operation was requested from a local connection, the source of this connection is a running process on the same system. The user running the process must have passed any authentication steps provided by the operating system. The user name of the owner of the process from which the connection was made is asserted as the identity. For example, this could be the name of the user running the shell from which an application has been started. The possible forms of authentication for local connections are:

1. Asserted user name (local OS)
2. Optional username and password (OS, LDAP or custom 3rd party repositories)
3. Security Token (JWT) IBM MQ only

If the administrative action was requested from a remote connection, then communications with IBM MQ are made through a network interface. The following forms of identity may be presented for authentication via network connections;

1. Asserted user name (from remote OS)
2. Username and password (OS, LDAP or custom 3rd party repositories)
3. Source network address (such as IP address)
4. X.509 Digital Certificate (mutual SSL/TLS authentication)
5. Security tokens (such as LTPA2 token or JWT token)
6. Other custom security (capability provided by 3rd party exits)
7. SSH keys

V 10.0.0 The IBM MQ Appliance Web UI can be configured to use OpenID Connect (OIDC) authentication.

IBM MQ's integration with IBM Cloud Pak® for Integration adds a new authentication type for the IBM MQ Console: Single Sign-On with the Cloud Pak. (CP4I only)

Role mapping:

In the role mapping stage, the credentials that were provided in the authentication stage may be mapped to an alternate user identifier. Provided the mapped user identifier is permitted to proceed (for example administrative users may be blocked by channel authentication rules), then the mapped user id is carried forward into the final stage when authorizing activities against IBM MQ resources.

Authorization:

IBM MQ provides the ability for different users to have different authorities against different messaging resources such as queues, topics and other queue manager objects.

Logging activity:

Some users of IBM MQ may need to create an audit record of access to MQ resources. Examples of desirable audit logs might include configuration changes that contain information about the change in addition to who requested it.

The following sources of information are available to implement this requirement:

1. An IBM MQ queue manager can be configured to produce command events when an admin command has been run successfully.
2. An IBM MQ queue manager can be configured to produce configuration events when a queue manager resource is created, altered or deleted.
3. An IBM MQ queue manager can be configured to produce an authority event when an authorization check fails for a resource.
4. Error messages indicating failed authorization checks are written to the queue manager error logs.
5. The IBM MQ Console will write audit messages to its logs when authentication, authorization checks fail or when queue managers are created, started, stopped or deleted.
6. The IBM MQ Appliance will write audit messages to its logs to record user log ins and system changes.
7. V 10.0.0 An IBM MQ queue manager can be configured to produce an authority event when an authorization check succeeds for a resource.
8. The IBM MQ Agent will write messages to its log files to record authorization failures.

When considering these kind of solutions, IBM MQ users might want to give consideration to the following points:

- Event messages are non-persistent so when a queue manager restarts the information is lost. Any event monitors should be configured to constantly consume any available messages and transfer the content to persistent media.
- IBM MQ privileged users have sufficient privileges to disabled events, clear logs or delete queue managers.

For more information about securing access to IBM MQ data and providing an audit trail refer to the following topics:

- IBM MQ security mechanisms
- Configuration events
- Command events
- Using error logs
- Security for the IBM MQ Agent

Data Processing

Encryption using a Public Key Infrastructure:

You can secure network connections to IBM MQ by specifying that the connections use TLS, which can also provide mutual authentication of the initiating side of the connection.

Using the PKI security facilities that are provided by transport mechanisms is the first step towards securing data processing with IBM MQ. However, without enabling further security features, the behavior of a consuming application is to process all messages delivered to it without validating the origin of the message or whether it was altered whilst in transit.

Users of IBM MQ that are licensed to use Advanced Message Security (AMS) capabilities can control the way in which applications process personal data held in messages, through the definition and configuration of security policies. Security policies allow digital signing and/or encryption to be applied to message data between applications.

It is possible to use security policies to require and validate a digital signature when consuming messages to ensure messages are authentic. AMS encryption provides a method by which message data is converted from a readable form to an encoded version that can only be decoded by another application if it is the intended recipient or the message and has access to the correct decryption key.

V 10.0.0 It is possible to ensure that replication traffic in a replicated data queue manager (RDQM) configuration is encrypted. This security can be configured when creating RDQM instances.

For more information about using SSL and certificates to secure your network connections, refer to the following topics in the IBM MQ product documentation:

- Configuring TLS security for IBM MQ
- AMS Overview
- V 10.0.0 Secure replication for HA RDQM, Secure replication for DR RDQM, Secure replication for DR/HA RDQM.

Data Deletion

IBM MQ provides commands and user interface actions to delete data that has been supplied to the product. This means that users of IBM MQ can delete data that relates to particular individuals, should this be required.

- Areas of IBM MQ behavior to consider for complying with GDPR Client Data deletion
  - Delete message data stored on an application queue by:
    - Removing individual messages using messaging API or tooling or by using message expiry.
    - Specifying that messages are non-persistent, held on a queue where non-persistent message class is normal and restarting the queue manager.
    - Administratively clearing the queue.
    - Deleting the queue.
  - Delete retained publication data stored on a topic by:
    - Specifying that messages are non-persistent and restarting the queue manager.
    - Replacing the retained data with new data or by using message expiry.
    - Administratively clearing the topic string.
  - Delete data stored on a queue manager by deleting the whole queue manager and any replicated copies for high availability or disaster recovery.
  - Delete data stored by the Service trace commands by deleting the files in the trace directory.
  - Delete FFST data stored by deleting the files in the errors directory.
  - Delete address space and Coupling Facility dumps (on z/OS).
  - Delete archive, backup or other copies of such data.
- Areas of IBM MQ behavior to consider for complying with GDPR Account Data deletion
  - You can delete account data and preferences stored by IBM MQ for connecting to queue managers and 3rd party services by deleting (including archive, backup or otherwise replicated copies thereof):
    - Queue manager authentication information objects that store credentials.
    - Queue manager authority records that reference user identifiers.
    - Queue manager channel authentication rules that map or block specific IP addresses, certificate DN's or user identifiers.
    - Credentials files used by IBM MQ Managed File Transfer Agent, Logger and MQ Explorer MFT Plugin for authentication with queue manager and file servers.
    - X.509 digital certificates that represent or contain information about an individual from keystores which may be used by SSL/TLS connections or IBM MQ Advanced Message Security (AMS).

- Individual user accounts from IBM MQ Appliance, including reference to those accounts in system log files.
- IBM MQ Explorer workspace metadata and Eclipse settings.
- IBM MQ Explorer password store as specified in the Password Preferences.
- IBM MQ Console and mqweb server configuration files.
- IBM MQ Internet Pass-Thru configuration files and keystores.
- IBM MQ Agent Secret objects, such as the CCDT secret.

Read more:

- MFT and IBM MQ connection authentication
- Mapping credentials for a file server by using the ProtocolBridgeCredentials.xml file
- Configuring IBM MQ Console users and roles

Data Monitoring

IBM MQ provides a range of monitoring features that users can exploit to gain a better understanding of how applications and queue managers are performing.

IBM MQ also provides a number a features that help manage queue manager error logs.

Read more:

- Monitoring your IBM MQ network
- Diagnostic message services
- QMErrorLog service
- Monitoring when using the IBM MQ Operator
- IBM MQ Appliance monitoring and reporting

Capability for Restricting Use of Personal Data

Using the facilities summarized in this document, IBM MQ enables an end-user to restrict usage of their personal data.

IBM MQ message queues should not be used as a permanent data store in the same way as a database, which is particularly true when handling application data that is subject to GDPR.

Unlike a database where data may be found through a search query, it can be difficult to find message data unless you know the queue, message and correlation identifiers of a message.

Provided messages containing an individual's data can be readily identified and located, it is possible using standard IBM MQ messaging features to access or modify message data.

File handling

1. IBM MQ Managed File Transfer does not perform malware scanning on files transferred. Files are transferred as-is and an integrity check is performed to ensure the file data is not modified during transfer. The source and destination checksums are published as part of transfer status publication. It is recommended that end users implement malware scanning as appropriate for their environment before MFT transfers the file and after MFT delivers a file to a remote end point.
2. IBM MQ Managed File Transfer does not take actions based on MIME type or file extension. MFT reads the files and transfers the bytes exactly as read from the input file.

Architectures based on a single queue manager

The simplest IBM MQ architectures involve the configuration and use of a single queue manager.

Before you plan your IBM MQ architecture, familiarize yourself with the basic IBM MQ concepts. See IBM MQ Technical overview.

A number of possible architectures using a single queue manager are described in the following sections:

- "Single queue manager with local applications accessing a service" on page 18
- "Single queue manager with remote applications accessing a service as clients" on page 18
- "Single queue manager with a publish/subscribe configuration" on page 18

Single queue manager with local applications accessing a service

The first architecture based on a single queue manager is where the applications accessing a service are running on the same system as the applications providing the service. An IBM MQ queue manager provides asynchronous intercommunication between the applications requesting the service and the applications providing the service. This means that communication between the applications can continue even if one of the applications is offline for an extended period of time.

Single queue manager with remote applications accessing a service as clients

The second architecture based on a single queue manager has the applications running remotely from the applications providing the service. The remote applications are running on different systems to the services. The applications connect as clients to the single queue manager. This means that access to a service can be provided to multiple systems through a single queue manager.

A limitation of this architecture is that a network connection must be available for an application to operate. The interaction between the application and the queue manager over the network connection is synchronous.

Single queue manager with a publish/subscribe configuration

An alternative architecture using a single queue manager is to use a publish/subscribe configuration. In publish/subscribe messaging, you can decouple the provider of information from the consumers of that information. This differs from the point to point messaging styles in the previously described architectures, where the applications must know information about the target application, for example the queue name to put messages on. Using IBM MQ publish/subscribe the sending application publishes a message with a specified topic based on the subject of the information. IBM MQ handles the distribution of the message to applications that have registered an interest in that subject through a subscription. The receiving applications also do not need to know anything about the source of the messages to receive them. For more information, see Publish/subscribe messaging and Example of a single queue manager publish/subscribe configuration.

Related concepts

Introduction to IBM MQ

Related tasks

"Planning an IBM MQ architecture" on page 5
When planning your IBM MQ environment, consider the support that IBM MQ provides for single and multiple queue manager architectures, and for point-to-point and publish/subscribe messaging styles. Also plan your resource requirements, and your use of logging and backup facilities.

Creating and managing queue managers on Multiplatforms

Architectures based on multiple queue managers

You can use distributed message queuing techniques to create an IBM MQ architecture involving the configuration and use of multiple queue managers.

Before you plan your IBM MQ architecture, familiarize yourself with the basic IBM MQ concepts. See IBM MQ Technical overview.

An IBM MQ architecture can be changed, without alteration to applications that provide services, by adding additional queue managers.

Applications can be hosted on the same machine as a queue manager, and then gain asynchronous communication with a service hosted on another queue manager on another system. Alternatively, applications accessing a service can connect as clients to a queue manager that then provides asynchronous access to the service on another queue manager.

Routes that connect different queue managers and their queues are defined using distributed queuing techniques. The queue managers within the architecture are connected using channels. Channels are used to move messages automatically from one queue manager to another in one direction depending on the configuration of the queue managers.

For a high level overview of planning an IBM MQ network, see “Designing distributed queue manager networks” on page 20.

For information about how to plan channels for your IBM MQ architecture, see IBM MQ distributed queuing techniques.

Distributed queue management enables you to create and monitor the communication between queue managers. For more information about distributed queue management, see Introduction to distributed queue management.

Related tasks

“Planning an IBM MQ architecture” on page 5
When planning your IBM MQ environment, consider the support that IBM MQ provides for single and multiple queue manager architectures, and for point-to-point and publish/subscribe messaging styles. Also plan your resource requirements, and your use of logging and backup facilities.

Creating and managing queue managers on Multiplatforms

Planning your distributed queues and clusters

You can manually connect queues hosted on distributed queue managers, or you can create a queue manager cluster and let the product connect the queue managers for you. To choose a suitable topology for your distributed messaging network, you need to consider your requirements for manual control, network size, frequency of change, availability and scalability.

Before you begin

This task assumes that you understand what distributed messaging networks are, and how they work. For a technical overview, see Distributed queuing and clusters.

About this task

To create a distributed messaging network, you can manually configure channels to connect queues hosted on different queue managers, or you can create a queue manager cluster. Clustering enables queue managers to communicate with each other without the need to set up extra channel definitions or remote queue definitions, simplifying their configuration and management.

To choose a suitable topology for your distributed publish/subscribe network, you need to consider the following broad questions:

- How much manual control do you need over the connections in your network?
- How big will your network be?

- How dynamic will it be?
- What are your availability and scalability requirements?

Procedure

- Consider how much manual control you need over the connections in your network.

  If you only need a few connections, or if individual connections need to be very precisely defined, you should probably create the network manually.

  If you need multiple queue managers that are logically related, and that need to share data and applications, you should consider grouping them together in a queue manager cluster.
- Estimate how big your network needs to be.
  a) Estimate how many queue managers you need. Bear in mind that queues can be hosted on more than one queue manager.
  b) If you are considering using a cluster, add two extra queue managers to act as full repositories.

  For larger networks, manual configuration and maintenance of connections can be very time consuming, and you should consider using a cluster.
- Consider how dynamic the network activity will be.

  Plan for busy queues to be hosted on performant queue managers.

  If you expect queues to be frequently created and deleted, consider using a cluster.
- Consider your availability and scalability requirements.
  a) Decide whether you need to guarantee high availability of queue managers. If so, estimate how many queue managers this requirement applies to.
  b) Consider whether some of your queue managers are less capable than others.
  c) Consider whether the communication links to some of your queue managers are more fragile than to others.
  d) Consider hosting queues on multiple queue managers.

  Manually configured networks and clusters can both be configured to be highly available and scalable. If you use a cluster, you need to define two extra queue managers as full repositories. Having two full repositories ensures that the cluster continues to operate if one of the full repositories becomes unavailable. Make sure that the full repository queue managers are robust, performant, and have good network connectivity. Do not plan to use the full repository queue managers for any other work.
- Based on these calculations, use the links provided to help you decide whether to manually configure connections between queue managers, or to use a cluster.

What to do next

You are now ready to configure your distributed messaging network.

Related tasks

Configuring distributed queuing

Configuring a queue manager cluster

Designing distributed queue manager networks

IBM MQ sends and receives data between applications, and over networks using Queue Managers and Channels. Network planning involves defining requirements to create a framework for connecting these systems over a network.

Channels can be created between your system and any other system with which you need to have communications. Multi-hop channels can be created to connect to systems where you have no direct connections. The message channel connections described in the scenarios are shown as a network diagram in Figure 1 on page 21.

If you need to create channels between systems on different physical networks, or channels that communicate through a firewall, using IBM MQ Internet Pass-Thru might simplify the configuration. For more information, see IBM MQ Internet Pass-Thru.

Channel and transmission queue names

Transmission queues can be given any name. But to avoid confusion, you can give them the same names as the destination queue manager names, or queue manager alias names, as appropriate. This associates the transmission queue with the route they use, giving a clear overview of parallel routes created through intermediate (multi-hopped) queue managers.

It is not so clear-cut for channel names. The channel names in Figure 1 on page 21 for QM2, for example, must be different for incoming and outgoing channels. All channel names can still contain their transmission queue names, but they must be qualified to make them unique.

For example, at QM2, there is a QM3 channel coming from QM1, and a QM3 channel going to QM3. To make the names unique, the first one might be named QM3_from_QM1, and the second might be named QM3_from_QM2. In this way, the channel names show the transmission queue name in the first part of the name. The direction and adjacent queue manager name are shown in the second part of the name.

A table of suggested channel names for Figure 1 on page 21 is given in Table 1 on page 21.

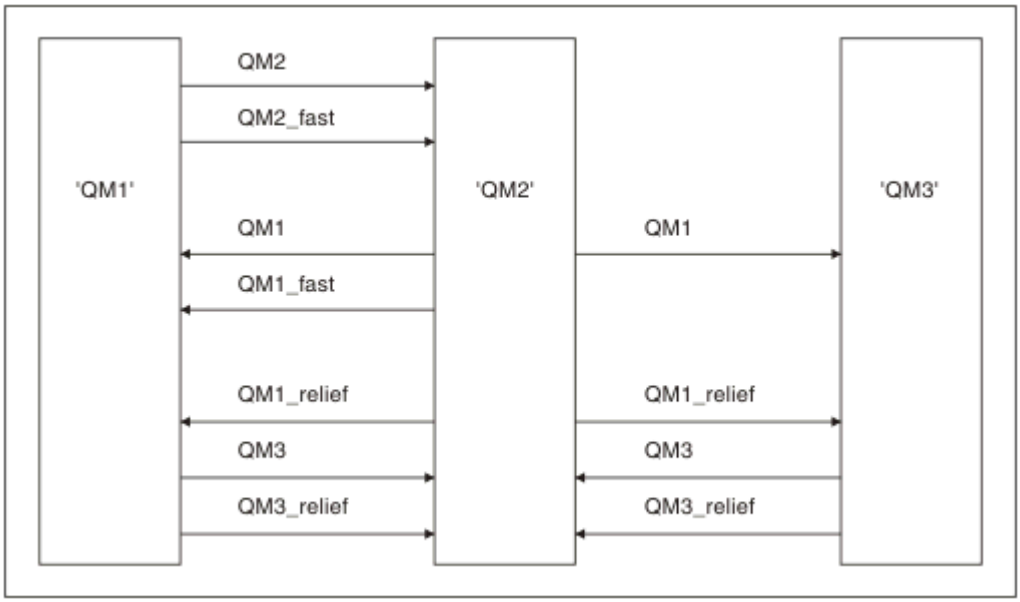


Figure 1. Network diagram showing all channels

Table 1. Example of channel names

Route name	Queue managers hosting channel	Transmission queue name	Suggested channel name
QM1	QM1 & QM2	QM1 (at QM2)	QM1.from.QM2
QM1	QM2 & QM3	QM1 (at QM3)	QM1.from.QM3
QM1_fast	QM1 & QM2	QM1_fast (at QM2)	QM1_fast.from.QM2
QM1_relief	QM1 & QM2	QM1_relief (at QM2)	QM1_relief.from.QM2
QM1_relief	QM2 & QM3	QM1_relief (at QM3)	QM1_relief.from.QM3
QM2	QM1 & QM2	QM2 (at QM1)	QM2.from.QM1
QM2_fast	QM1 & QM2	QM2_fast (at QM1)	QM2_fast.from.QM1

Table 1. Example of channel names (continued)			
Route name	**Queue managers hosting channel**	**Transmission queue name**	**Suggested channel name**
QM3	QM1 & QM2	QM3 (at QM1)	QM3.from.QM1
QM3	QM2 & QM3	QM3 (at QM2)	QM3.from.QM2
QM3_relief	QM1 & QM2	QM3_relief (at QM1)	QM3_relief.from.QM1
QM3_relief	QM2 & QM3	QM3_relief (at QM2)	QM3_relief.from.QM2

Note:

1. z/OS On IBM MQ for z/OS, queue manager names are limited to four characters.
2. Name all the channels in your network uniquely. As shown in Table 1 on page 21, including the source and target queue manager names in the channel name is a good way to do so.

Network planner

Creating a network assumes that there is another, higher level function of *network planner* whose plans are implemented by the other members of the team.

For widely used applications, it is more economical to think in terms of local access sites for the concentration of message traffic, using wide-band links between the local access sites, as shown in Figure 2 on page 23.

In this example there are two main systems and a number of satellite systems. The actual configuration would depend on business considerations. There are two concentrator queue managers located at convenient centers. Each QM-concentrator has message channels to the local queue managers:

- QM-concentrator 1 has message channels to each of the three local queue managers, QM1, QM2, and QM3. The applications using these queue managers can communicate with each other through the QM-concentrators.
- QM-concentrator 2 has message channels to each of the three local queue managers, QM4, QM5, and QM6. The applications using these queue managers can communicate with each other through the QM-concentrators.
- The QM-concentrators have message channels between themselves thus allowing any application at a queue manager to exchange messages with any other application at another queue manager.

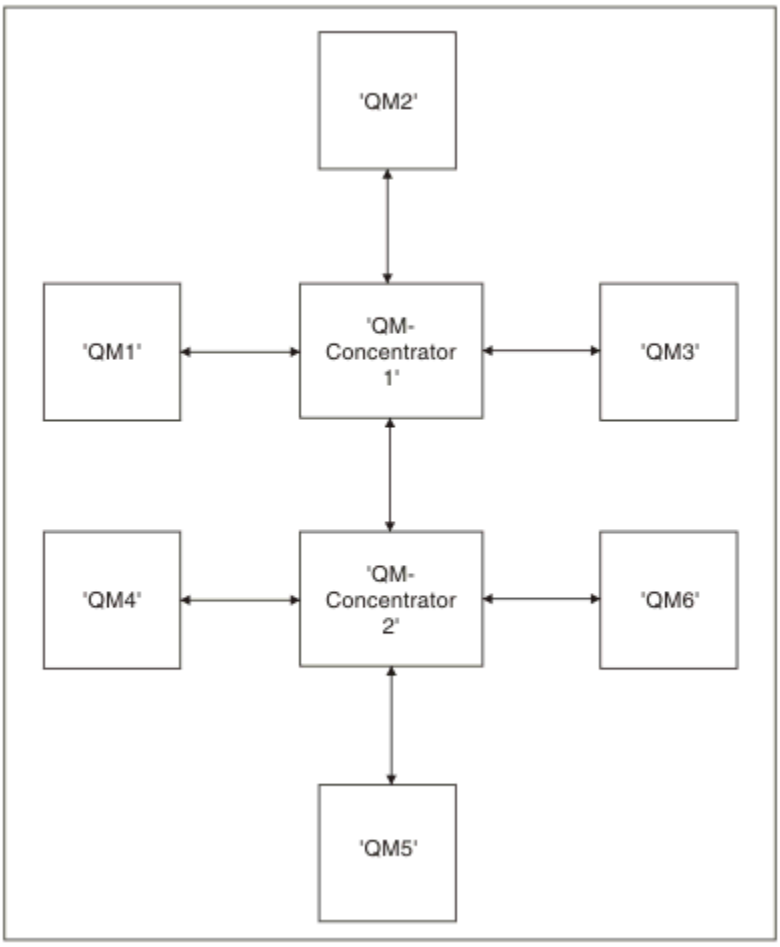


Figure 2. Network diagram showing QM-concentrators

Designing clusters

Clusters provide a mechanism for interconnecting queue managers in a way that simplifies both the initial configuration and the ongoing management. Clusters must be carefully designed, to ensure that they function correctly, and that they achieve the required levels of availability and responsiveness.

Before you begin

For an introduction to clustering concepts, see the following topics:

- Distributed queuing and clusters
- "Comparison of clustering and distributed queuing" on page 29
- Components of a cluster

When you are designing your queue manager cluster you have to make some decisions. You must first decide which queue managers in the cluster are to hold the full repositories of cluster information. Any queue manager you create can work in a cluster. You can choose any number of queue managers for this purpose but the ideal number is two. For information about selecting queue managers to hold the full repositories, see "How to choose cluster queue managers to hold full repositories" on page 31.

See the following topics for more information about designing your cluster:

- "Example clusters" on page 37
- "Organizing a cluster" on page 33
- "Cluster naming conventions" on page 33

- z/OS "Queue sharing groups and clusters" on page 34
- "Overlapping clusters" on page 35

What to do next

See the following topics for more information about configuring and working with clusters:

- Establishing communication in a cluster
- Configuring a queue manager cluster
- Routing messages to and from clusters
- Using clusters for workload management

For more information to help you configure your cluster, see "Clustering tips" on page 36.

Planning how you use multiple cluster transmission queues

You can explicitly define transmission queues, or have the system generate the transmission queues for you. If you define the transmission queues yourself, you have more control over the queue definitions.

z/OS On z/OS, you also have more control over the page set where the messages are held.

Defining the transmission queues

There are two methods of defining transmission queues:

- Automatically, using the queue manager attribute DEFCLXQ, as follows:

  ```
  ALTER QMGR DEFCLXQ(SCTQ | CHANNEL)
  ```

  DEFCLXQ(SCTQ) indicates that the default transmission queue for all cluster-sender channels is SYSTEM.CLUSTER.TRANSMIT.QUEUE. This is the default value.

  DEFCLXQ(CHANNEL) indicates that by default each cluster-sender channel uses a separate transmission queue named SYSTEM.CLUSTER.TRANSMIT.*channel name*. Each transmission queue is automatically defined by the queue manager. See "Automatically-defined cluster transmission queues" on page 25 for more information.
- Manually, by defining a transmission queue with a value specified for the CLCHNAME attribute. The CLCHNAME attribute indicates which cluster-sender channels should use the transmission queue.

  z/OS If you are manually defining a transmission queue on z/OS, see "Planning for manually-defined cluster transmission queues" on page 27 for more information.

What security do I need?

To initiate a switch, either automatically or manually, you need authority to start a channel.

To define the queue used as a transmission queue, you need standard IBM MQ authority to define the queue.

When is a suitable time to implement the change?

When changing the transmission queue used by cluster-sender channels, you need to allocate a time in which to make the update, considering the following points:

- The time required for a channel to switch transmission queue depends on the total number of messages on the old transmission queue, how many messages need to be moved, and the size of the messages.
- Applications can continue to put messages to the transmission queue while the change is happening. This might lead to an increase in the transition time.
- You can change the CLCHNAME parameter of any transmission queue or DEFCLXQ at any time, preferably when the workload is low.

Note that nothing happens immediately.

- Changes occur only when a channel starts or restarts. When a channel starts it checks the current configuration and switches to a new transmission queue if required.
- There are several changes that might alter the association of a cluster-sender channel with a transmission queue:
  - Altering the value of a transmission queue's CLCHNAME attribute, making CLCHNAME less specific or blank.
  - Altering the value of a transmission queue's CLCHNAME attribute, making CLCHNAME more specific.
  - Deleting a queue with CLCHNAME specified.
  - Altering the queue manager attribute DEFCLXQ.

How long will the switch take?

During the transition period, any messages for the channel are moved from one transmission queue to another. The time required for a channel to switch transmission queue depends on the total number of messages on the old transmission queue, and how many messages need to be moved.

For queues containing a few thousand messages, it should take under a second to move the messages. The actual time depends on the number and size of the messages. Your queue manager should be able to move messages at many megabytes each second.

Applications can continue to put messages to the transmission queue while the change is happening. This might lead to an increase in the transition time.

Each affected cluster-sender channel must be restarted for the change to take effect. Therefore, it is best to change the transmission queue configuration when the queue manager is not busy, and few messages are stored on the cluster transmission queues.

The **runswchl** command z/OS or the SWITCH CHANNEL(*) STATUS command in CSQUTIL on z/OS can be used to query the status of cluster-sender channels and what pending changes are outstanding to their transmission queue configuration.

How to implement the change

See Implementing the system using multiple cluster transmission queues for details on how you make the change to multiple cluster transmission queues, either automatically or manually.

Undoing the change

z/OS

See Undoing a change to a transmission queue on z/OS for details on how you back out changes if you encounter problems.

Automatically-defined cluster transmission queues
You can have the system generate the transmission queues for you.

Before you begin

z/OS To set up the cluster transmission queues manually on z/OS, see “Planning for manually-defined cluster transmission queues” on page 27.

About this task

If a channel does not have a manually defined cluster transmission queue that is associated with it, and you specify DEFCLXQ(CHANNEL), when the channel starts the queue manager automatically defines a permanent-dynamic queue for the cluster sender channel. Model queue

SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE is used to automatically define the permanent dynamic cluster transmit queue with the name SYSTEM.CLUSTER.TRANSMIT.ChannelName.

Important: z/OS In IBM MQ 8.0, the queue manager does not have the SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE. You cannot migrate directly from IBM MQ 8.0 to this version. For information about adding the SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE into a queue manager that is migrated from IBM MQ 8.0, see this topic in the documentation for the interim version that you used to migrate the queue manager.

Procedure

1. Use the *DEFCLXQ* queue manager attribute.

   For more information on this attribute, see ALTER QMGR.

   There are two options:

   SCTQ
   This option is the default, and means that you use the single SYSTEM.CLUSTER.TRANSMIT.QUEUE.

   CHANNEL
   Means that you use multiple cluster transmission queues.
2. To switch to the new association:
   - Stop and restart the channel.
   - The channel uses the new transmission queue definition.
   - Messages are transferred by a transitional switch process from the old queue to the new transmission queue.

   Note that any application messages are put to the old definition.

   When the number of messages on the old queue reaches zero, new messages are placed directly on the new transmission queue.
3. To monitor when the switching process finishes:
   a) A switch of transmission queue that is initiated by a channel runs in the background and your administrator can monitor the queue manager job log to determine when it has completed.
   b) Monitor messages on the job log to show the progress of the switch.
   c) To make sure that only the channels that you wanted are using this transmission queue, issue the command DIS CLUSQMGR(*) where, for example, the transmission queue property that defines the transmission queue is APPQMGR.CLUSTER1.XMITQ.
   d) z/OS
   Use the SWITCH CHANNEL (*) STATUS command under CSQUTIL.

   This option tells you what pending changes are outstanding, and how many messages need to be moved between transmission queues.

Results

You have set up your cluster transmission queue, or queues.

Related tasks

“Planning for manually-defined cluster transmission queues” on page 27
On IBM MQ for z/OS, if you define the transmission queues yourself you have more control over the definitions, and the page set on which the messages are held.

Related reference

ALTER QMGR
DISPLAY CLUSQMGR

z/OS *Planning for manually-defined cluster transmission queues*

On IBM MQ for z/OS, if you define the transmission queues yourself you have more control over the definitions, and the page set on which the messages are held.

Before you begin

To set up cluster transmission queues automatically, see "Automatically-defined cluster transmission queues" on page 25.

About this task

Your administrator manually defines a transmission queue and uses queue attribute CLCHNAME to define which cluster sender channel, or channels, will use this queue as their transmission queue.

Note that CLCHNAME can include a wildcard character at the beginning, or end, to allow a single queue to be used for multiple channels.

Procedure

1. For example, enter the following:

```
DEFINE QLOCAL(APPQMGR.CLUSTER1.XMITQ)
CLCHNAME(CLUSTER1.TO.APPQMGR)
USAGE(XMITQ) STGCLASS(STG1)
INDXTYPE( CORRELID ) SHARE

DEFINE STGCLASS(STG1) PSID(3)
DEFINE PSID(3) BUFFERPOOL(4)
```

   Tip: You need to plan which page set (and buffer pool) you use for your transmission queues. You can have different page sets for different queues, and provide isolation between them, so one page set filling up does not impact transmission queues in other page sets.

   See Working with cluster transmission queues and cluster-sender channels for information on how each channel selects the appropriate queue.

   When the channel starts it switches its association to the new transmission queue. To make sure no message is lost, the queue manager automatically transfers messages from the old cluster transmission queue to the new transmission queue in order.

2. Use the CSQUTIL SWITCH function to change to the new association.

   See Switch the transmission queue associated with cluster-sender channels (SWITCH) for further information.

   a) STOP the channel, or channels, whose transmission queue is to be changed, so that they are in STOPPED status.

      For example:

```
STOP CHANNEL(CLUSTER1.TO.APPQMGR)
```

   b) Change the CLCHNAME(XXXX) attribute on the transmission queue.

   c) Use the SWITCH function to switch the messages or monitor what is happening.

      Use the command

```
SWITCH CHANNEL(*) MOVEMSGS(YES)
```

      to move the messages without starting the channel.

   d) Start the channel, or channels, and check whether the channel is using the correct queues.

      For example:

```
DIS CHS(CLUSTER1.TO.APPQMGR)
DIS CHS(*) where(XMITQ eq APPQMGR.CLUSTER1.XMITQ)
```

Tip: The following process uses the CSQUTIL SWITCH function. For more information, see Switch the transmission queue associated with cluster-sender channels (SWITCH).

You do not have to use this function but using this function gives more options:

- Using SWITCH CHANNEL (*) STATUS provides an easy way to identify the switching status of cluster-sender channels. It allows your administrator to see what channels are currently switching, and those channels having a switch pending that take effect when those channels next start.

  Without this capability your administrator needs to use multiple DISPLAY commands, and then process the resulting output to ascertain this information. Your administrator can also confirm that a configuration change has the required result.
- If CSQUTIL is used to initiate the switch, CSQUTIL continues to monitor the progress of this operation, and only ends when the switch has completed.

  This can make it much easier to perform these operations in batch. Also, if CSQUTIL is run to switch multiple channels, CSQUTIL performs these actions sequentially; this can have less impact for your enterprise than multiple switches running in parallel.

Results

You have set up your cluster transmission queue, or queues, on z/OS.

Access control and multiple cluster transmission queues

Choose between three modes of checking when an application puts messages to remote cluster queues. The modes are checking remotely against the cluster queue, checking locally against `SYSTEM.CLUSTER.TRANSMIT.QUEUE`, or checking against local profiles for the cluster queue, or cluster queue manager.

IBM MQ gives you the choice of checking locally, or locally and remotely, that a user has permission to put a message to a remote queue. A typical IBM MQ application uses local checking only, and relies on the remote queue manager trusting the access checks made on the local queue manager. If remote checking is not used, the message is put to the target queue with the authority of the remote message channel process. To use remote checking you must set the put authority of the receiving channel to context security.

The local checks are made against the queue that the application opens. In distributed queuing, the application usually opens a remote queue definition, and access checks are made against the remote queue definition. If the message is put with a full routing header, the checks are made against the transmission queue. If an application opens a cluster queue that is not on the local queue manager, there is no local object to check. The access control checks are made against the cluster transmission queue, `SYSTEM.CLUSTER.TRANSMIT.QUEUE`. Even with multiple cluster transmission queues, local access control checks for remote cluster queues are made against `SYSTEM.CLUSTER.TRANSMIT.QUEUE`.

The choice of local or remote checking is a choice between two extremes. Checking remotely is fine-grained. Every user must have an access control profile on every queue manager in the cluster to put to any cluster queue. Checking locally is coarse-grained. Every user needs only one access control profile for the cluster transmission queue on the queue manager they are connected to. With that profile, they can put a message to any cluster queue on any queue manager in any cluster.

Administrators have another way to set up access control for cluster queues. You can create a security profile for a cluster queue on any queue manager in the cluster using the **setmqaut** command. The profile takes effect if you open a remote cluster queue locally, specifying only the queue name. You can also set up a profile for a remote queue manager. If you do so, the queue manager can check the profile of a user that opens a cluster queue by providing a fully qualified name.

The new profiles work only if you change the queue manager stanza **ClusterQueueAccessControl** setting to `RQMName`. The default is `Xmitq`. Before you modify this setting, ensure that you create profiles for all existing applications that access remote cluster queues. If you change the stanza setting to `RQMName` without creating profiles, the applications are likely to fail.

Tip: Cluster queue accessing checking does not apply to remote queuing. Access checks are still made against local definitions. The changes mean that you can follow the same approach to configure access checking on cluster queues and cluster topics. z/OS The changes also align the access checking approach for cluster queues more closely with z/OS. The commands to set up access checking on z/OS are different, but both check access against a profile rather than against the object itself.

Related concepts

"Clustering: Application isolation using multiple cluster transmission queues" on page 45
You can isolate the message flows between queue managers in a cluster. You can place messages being transported by different cluster-sender channels onto different cluster transmission queues. You can use the approach in a single cluster or with overlapping clusters. The topic provides examples and some best practices to guide you in choosing an approach to use.

Related tasks

Setting `ClusterQueueAccessControl`

Comparison of clustering and distributed queuing

Compare the components that need to be defined to connect queue managers using distributed queuing and clustering.

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, you must define:

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

Figure 3 on page 29 shows the components required for distributed queuing.

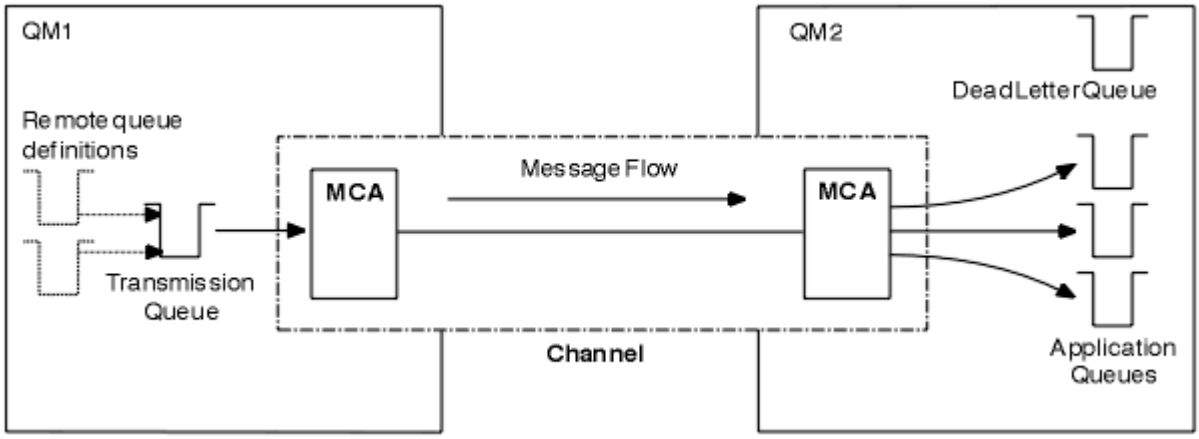


Figure 3. Distributed queuing

If you group queue managers in a cluster, queues on any queue manager are available to any other queue manager in the cluster. Any queue manager can send a message to any other queue manager in the same cluster without explicit definitions. You do not provide 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

Definitions to set up a cluster versus distributed queuing

Look at Figure 4 on page 30, which shows four queue managers each with two queues. Consider how many definitions are needed to connect these queue managers using distributed queuing. Compare how many definitions are needed to set up the same network as a cluster.

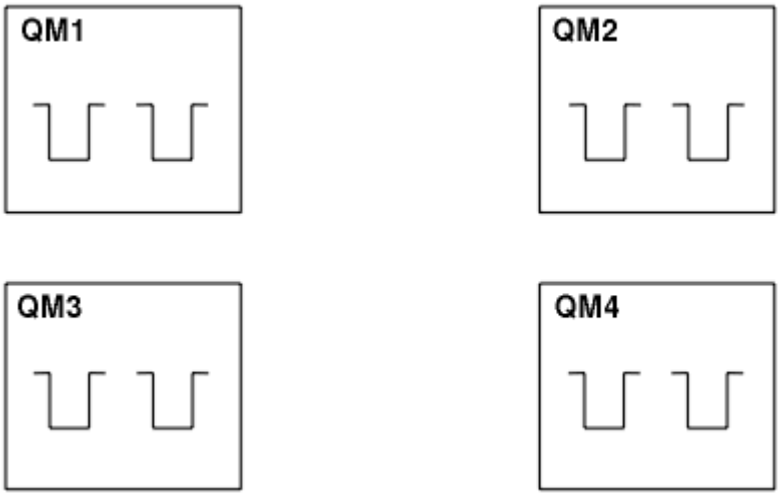


Figure 4. A network of four queue managers

Definitions to set up a network using distributed queuing

To set up the network shown in Figure 3 on page 29 using distributed queuing, you might have the following definitions:

Table 2. Definitions for distributed queuing

Description	Number per queue manager	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

You might reduce this number of definitions by using generic receiver-channel definitions. The maximum number of definitions could be as many as 17 on each queue manager, which is a total of 68 for this network.

Definitions to set up a network using clusters

To set up the network shown in Figure 3 on page 29 using clusters you need the following definitions:

Table 3. Definitions for clustering

Description	Number per queue manager	Total number
A cluster-sender channel definition for a channel on which to send messages to a repository queue manager	1	4

Table 3. Definitions for clustering (continued)		
Description	**Number per queue manager**	**Total number**
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

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

Only one CLUSSDR and one CLUSRCVR channel definition is required. When the cluster is defined, you can add or remove queue managers (other than the repository queue managers) without any disruption to the other queue managers.

Using a cluster 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, you can move cluster queues from one queue manager to another within the cluster without having to do 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.

How to choose cluster queue managers to hold full repositories

In each cluster you must choose at least one, and preferably two queue managers to hold full repositories. Two full repositories are sufficient for all but the most exceptional circumstances. If possible, choose queue managers that are hosted on robust and permanently-connected platforms, that do not have coinciding outages, and that are in a central position geographically. Also consider dedicating systems as full repository hosts, and not using these systems for any other tasks.

Full repositories are queue managers that maintain a complete picture of the state of the cluster. To share this information, each full repository is connected by CLUSSDR channels (and their corresponding CLUSRCVR definitions) to every other full repository in the cluster. You must manually define these channels.

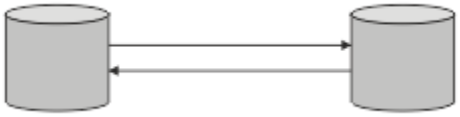


Figure 5. Two connected full repositories.

Every other queue manager in the cluster maintains a picture of what it currently knows about the state of the cluster in a *partial repository*. These queue managers publish information about themselves, and request information about other queue managers, using any two available full repositories. If a chosen full repository is not available, another is used. When the chosen full repository becomes available again, it collects the latest new and changed information from the others so that they keep in step. If all the full repositories go out of service, the other queue managers use the information they have in their partial repositories. However, they are limited to using the information that they have; new information and

requests for updates cannot be processed. When the full repositories reconnect to the network, messages are exchanged to bring all repositories (both full and partial) up to date.

When planning the allocation of full repositories, include the following considerations:

- The queue managers chosen to hold full repositories need to be reliable and managed. Choose queue managers that are hosted on a robust and permanently-connected platform.
- Consider the planned outages for the systems hosting your full repositories, and ensure that they do not have coinciding outages.
- Consider network performance: Choose queue managers that are in a central position geographically, or that share the same system as other queue managers in the cluster.
- Consider whether a queue manager is a member of more than one cluster. It can be administratively convenient to use the same queue manager to host the full repositories for several clusters, provided this benefit is balanced against how busy you expect the queue manager to be.
- Consider dedicating some systems to contain only full repositories, and not using these systems for any other tasks. This ensures that these systems only require maintenance for queue manager configuration, and are not removed from service for the maintenance of other business applications. It also ensures that the task of maintaining the repository does not compete with applications for system resources. This can be particularly beneficial in large clusters (say, clusters of more than a thousand queue managers), where full repositories have a much higher workload in maintaining the cluster state.

Having more than two full repositories is possible, but rarely recommended. Although object definitions (that is, queues, topics and channels) flow to all available full repositories, requests only flow from a partial repository to a maximum of two full repositories. This means that, when more than two full repositories are defined, and any two full repositories become unavailable, some partial repositories might not receive updates they would expect. See MQ Clusters: Why only two Full Repositories?

One situation in which you might find it useful to define more than two full repositories is when migrating existing full repositories to new hardware or new queue managers. In this case, you should introduce the replacement full repositories, and confirm that they have become fully populated, before you remove the previous full repositories. Whenever you add a full repository, remember that you must directly connect it to every other full repository with CLUSSDR channels.

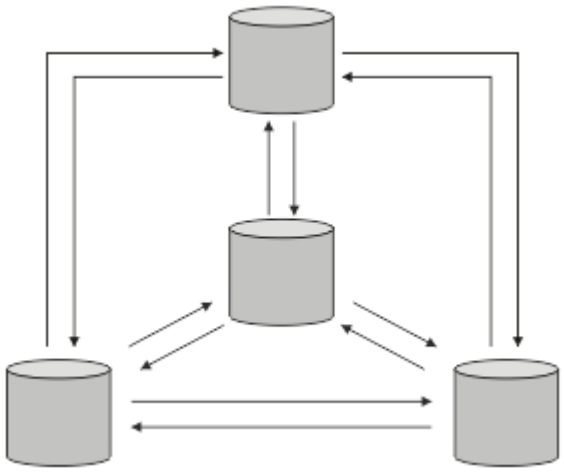


Figure 6. More than two connected full repositories

Related information

MQ Clusters: Why only two Full Repositories?
How big can an MQ Cluster be?

Organizing a cluster

Select which queue managers to link to which full repository. Consider the performance effect, the queue manager version, and whether multiple CLUSSDR channels are desirable.

Having selected the queue managers to hold full repositories, you need to decide which queue managers to link to which full repository. The CLUSSDR channel definition links a queue manager to a full repository from which it finds out about the other full repositories in the cluster. From then on, the queue manager sends messages to any two full repositories. It always tries to use the one to which it has a CLUSSDR channel definition first. You can choose to link a queue manager to either full repository. In choosing, consider the topology of your configuration, and the physical or geographical location of the queue managers.

Because all cluster information is sent to two full repositories, there might be situations in which you want to make a second CLUSSDR channel definition. You might define a second CLUSSDR channel in a cluster that has many full repositories spread over a wide area. You can then control which two full repositories your information is sent to.

Cluster naming conventions

Consider naming queue managers in the same cluster using a naming convention that identifies the cluster to which the queue manager belongs. Use a similar naming convention for channel names and extend it to describe the channel characteristics.

Best practices when naming MQ Clusters

Although cluster names can be up to 48 characters, relatively short cluster names are helpful when applying naming conventions to other objects. See “Best practices when choosing cluster channel names” on page 33.

When choosing a cluster name, it is usually helpful to represent the 'purpose' of the cluster (which is likely to be long-lived) rather than the 'content'. For example 'B2BPROD' or 'ACTTEST' rather than 'QM1_QM2_QM3_CLUS'.

Best practices when choosing cluster Queue Manager names

If you are creating a new cluster and its members from scratch, consider a naming convention for the queue managers that reflects their cluster usage. Every queue manager must have a different name. However, you can give queue managers in a cluster a set of similar names, to help in identifying and remembering logical groupings (for example 'ACTTQM1, ACTTQM2).

Relatively short queue manager names (for example less than 8 characters) help if you choose to use the convention described in the next section, or something similar, for channel names.

Best practices when choosing cluster channel names

Because queue managers and clusters can have names of up to 48 characters, and a channel name is limited to 20 characters, take care when first naming objects to avoid having to change the naming convention midway through a project (see previous section).

When defining channels, remember that automatically-created cluster-sender channels on any queue manager in the cluster take their name from the corresponding cluster-receiver channel configured on the receiving queue manager in the cluster, and these must therefore be unique and make sense *on remote queue managers in the cluster.*

One common approach is to use the queue manager name preceded by the cluster name. For example, if the cluster name is CLUSTER1 and the queue managers are QM1, QM2, then cluster-receiver channels are CLUSTER1.QM1, CLUSTER1.QM2.

You might extend this convention if channels have different priorities or use different protocols. For example:

- CLUSTER1.QM1.S1

- CLUSTER1.QM1.N3
- CLUSTER1.QM1.T4

In this example, S1 might be the first SNA channel, N3 might be the NetBIOS channel with a network priority of three, and T4 might be TCP IP using an IPV4 network.

Naming shared channel definitions

A single channel definition can be shared across multiple clusters, in which case the naming conventions suggested here would need modification. However, as described in Managing channel definitions it is usually preferable to define discrete channels for each cluster in any case.

Older channel naming conventions

Outside of cluster environments it has historically been common to use a 'FROMQM.TO.TARGETQM' naming convention, so you might find existing clusters have used something similar (such as CLUSTER.TO.TARGET). This is not recommended as part of a new cluster naming scheme because it further reduces the available characters to convey 'useful' information within your channel name.

z/OS **Channel names on IBM MQ for z/OS**

You can define VTAM generic resources or *Dynamic Domain Name Server* (DDNS) generic names. You can define connection names using generic names. However, when you create a cluster-receiver definition, do not use a generic connection name.

The problem with using generic connection names for cluster-receiver definitions is as follows: If you define a CLUSRCVR with a generic CONNAME there is no guarantee that your CLUSSDR channels point to the queue managers you intend. Your initial CLUSSDR might end up pointing to any queue manager in the queue sharing group, not necessarily one that hosts a full repository. If a channel starts trying a connection again, it might reconnect to a different queue manager with the same generic name, disrupting the flow of messages.

z/OS *Queue sharing groups and clusters*

Shared queues can be cluster queues and queue managers in a queue sharing group can also be cluster queue managers.

On IBM MQ for z/OS you can group queue managers into queue sharing groups. A queue manager in a queue sharing group can define a local queue that is to be shared by up to 32 queue managers.

Shared queues can also be cluster queues. Furthermore, the queue managers in a queue sharing group can also be in one or more clusters.

You can define VTAM generic resources or *Dynamic Domain Name Server* (DDNS) generic names. You can define connection names using generic names. However, when you create a cluster-receiver definition, do not use a generic connection name.

The problem with using generic connection names for cluster-receiver definitions is as follows: If you define a CLUSRCVR with a generic CONNAME there is no guarantee that your CLUSSDR channels point to the queue managers you intend. Your initial CLUSSDR might end up pointing to any queue manager in the queue sharing group, not necessarily one that hosts a full repository. If a channel starts trying a connection again, it might reconnect to a different queue manager with the same generic name, disrupting the flow of messages.

A CLUSRCVR channel that uses the group listener port can not be started because, if this were the case, it would not be possible to tell which queue manager the CLUSRCVR would connect to each time. The cluster system queues on which information is kept about the cluster are not shared. Each queue manager has its own.

Cluster channels are used not only to transfer application messages but internal system messages about the setup of the cluster. Each queue manager in the cluster must receive these internal system messages to participate properly in clustering, so needs its own unique CLUSRCVR channel on which to receive them.

A shared CLUSRCVR could start on any queue manager in the queue sharing group (QSG) and so lead to an inconsistent supply of the internal system messages to the QSG queue managers, meaning none can properly participate in the cluster. To ensure no shared CLUSRCVR channels can be used, any attempt fails with the CSQX502E message.

Overlapping clusters

Overlapping clusters provide additional administrative capabilities. Use namelists to reduce the number of commands needed to administer overlapping clusters.

You can create clusters that overlap. There are a number of reasons you might define overlapping clusters; for example:

- To allow different organizations to have their own administration.
- To allow independent applications to be administered separately.
- To create classes of service.

In Figure 7 on page 35, the queue manager STF2 is a member of both the clusters. When a queue manager is a member of more than one cluster, you can take advantage of namelists to reduce the number of definitions you need. Namelists contain a list of names, for example, cluster names. You can create a namelist naming the clusters. Specify the namelist on the ALTER QMGR command for STF2 to make it a full repository queue manager for both clusters.

If you have more than one cluster in your network, you must give them different names. If two clusters with the same name are ever merged, it is not possible to separate them again. It is also a good idea to give the clusters and channels different names. They are more easily distinguished when you look at the output from the DISPLAY commands. Queue manager names must be unique within a cluster for it to work correctly.

Defining classes of service

Imagine a university that has a queue manager for each member of staff and each student. Messages between members of staff are to travel on channels with a high priority and high bandwidth. Messages between students are to travel on cheaper, slower channels. You can set up this network using traditional distributed queuing techniques. IBM MQ selects which channels to use by looking at the destination queue name and queue manager name.

To clearly differentiate between the staff and students, you could group their queue managers into two clusters as shown in Figure 7 on page 35. IBM MQ moves messages to the meetings queue in the staff cluster only over channels that are defined in that cluster. Messages for the gossip queue in the students cluster go over channels defined in that cluster and receive the appropriate class of service.

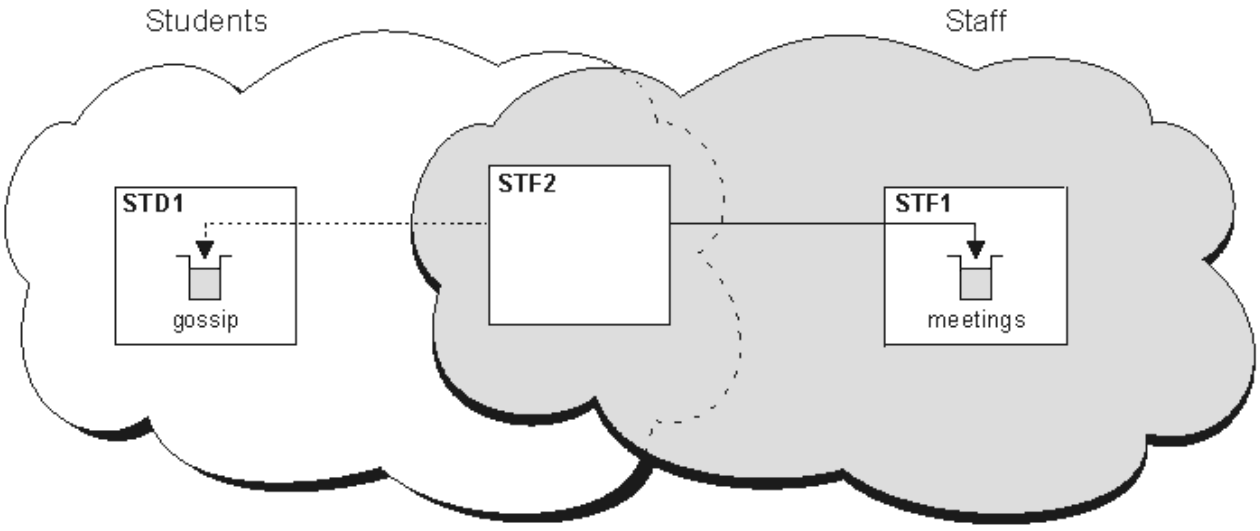


Figure 7. Classes of service

Clustering tips

You might need to make some changes to your systems or applications before using clustering. There are both similarities and differences from the behavior of distributed queuing.

- You must add manual configuration definitions to queue managers outside a cluster for them to access cluster queues.
- If you merge two clusters with the same name, you cannot separate them again. Therefore it is advisable to give all clusters a unique name.
- If a message arrives at a queue manager but there is no queue there to receive it, the message is put on the dead-letter queue. If there is no dead-letter queue, the channel fails and tries again. The use of the dead-letter queue is the same as with distributed queuing.
- The integrity of persistent messages is maintained. Messages are not duplicated or lost as a result of using clusters.
- Using clusters reduces system administration. Clusters make it easy to connect larger networks with many more queue managers than you would be able to contemplate using distributed queuing. There is a risk that you might consume excessive network resources if you attempt to enable communication between every queue manager in a cluster.
- If you use the IBM MQ Explorer, which presents the queue managers in a tree structure, the view for large clusters might be cumbersome.
- Multi The purpose of distribution lists is to use a single MQPUT command to send the same message to multiple destinations. Distribution lists are supported on IBM MQ for Multiplatforms. You can use distribution lists with queue manager clusters. In a cluster, all the messages are expanded at MQPUT time. The advantage, in terms of network traffic, is not so great as in a non-clustering environment. The advantage of distribution lists is that the numerous channels and transmission queues do not need to be defined manually.
- If you are going to use clusters to balance your workload examine your applications. See whether they require messages to be processed by a particular queue manager or in a particular sequence. Such applications are said to have message affinities. You might need to modify your applications before you can use them in complex clusters.
- You might choose to use the MQOO_BIND_ON_OPEN option on an MQOPEN to force messages to be sent to a specific destination. If the destination queue manager is not available the messages are not delivered until the queue manager becomes available again. Messages are not routed to another queue manager because of the risk of duplication.
- If a queue manager is to host a cluster repository, you need to know its host name or IP address. You have to specify this information in the CONNAME parameter when you make the CLUSSDR definition on other queue managers joining the cluster. If you use DHCP, the IP address is subject to change because DHCP can allocate a new IP address each time you restart a system. Therefore, you must not specify the IP address in the CLUSSDR definitions. Even if all the CLUSSDR definitions specify the host name rather than the IP address, the definitions would still not be reliable. DHCP does not necessarily update the DNS directory entry for the host with the new address. If you must nominate queue managers as full repositories on systems that use DHCP, install software that guarantees to keep your DNS directory up to date.
- Do not use generic names, for example VTAM generic resources or Dynamic Domain Name Server (DDNS) generic names as the connection names for your channels. If you do, your channels might connect to a different queue manager than expected.
- You can only get a message from a local cluster queue, but you can put a message to any queue in a cluster. If you open a queue to use the MQGET command, the queue manager opens the local queue.
- You do not need to alter any of your applications if you set up a simple IBM MQ cluster. The application can name the target queue on the MQOPEN call and does not need to know about the location of the queue manager. If you set up a cluster for workload management you must review your applications and modify them as necessary.
- You can view current monitoring and status data for a channel or queue using the DISPLAY CHSTATUS and the DISPLAY QSTATUS **runmqsc** commands. The monitoring information can be used to help

gauge the performance and health of the system. Monitoring is controlled by queue manager, queue, and channel attributes. Monitoring of auto-defined cluster-sender channels is possible with the MONACLS queue manager attribute.

Related concepts

Clusters

"Comparison of clustering and distributed queuing" on page 29
Compare the components that need to be defined to connect queue managers using distributed queuing and clustering.

Components of a cluster

Related tasks

Configuring a queue manager cluster

Setting up a new cluster

How long do the queue manager repositories retain information?

Queue manager repositories retain information for 30 days. An automatic process efficiently refreshes information that is being used.

When a queue manager sends out some information about itself, the full and partial repository queue managers store the information for 30 days. Information is sent out, for example, when a queue manager advertises the creation of a new queue. To prevent this information from expiring, queue managers automatically resend all information about themselves after 27 days. If a partial repository sends a new request for information part way through the 30 day lifetime, the expiry time remains the original 30 days.

When information expires, it is not immediately removed from the repository. Instead it is held for a grace period of 60 days. If no update is received within the grace period, the information is removed. The grace period allows for the fact that a queue manager might have been temporarily out of service at the expiry date. If a queue manager becomes disconnected from a cluster for more than 90 days, it stops being part of the cluster. However, if it reconnects to the network it becomes part of the cluster again. Full repositories do not use information that has expired to satisfy new requests from other queue managers.

Similarly, when a queue manager sends a request for up-to-date information from a full repository, the request lasts for 30 days. After 27 days IBM MQ checks the request. If it has been referenced during the 27 days, it is refreshed automatically. If not, it is left to expire and is refreshed by the queue manager if it is needed again. Expiring requests prevents a buildup of requests for information from dormant queue managers.

Note: You should download and install the PTF for APAR PH43191, which fixes system errors in calculating the expiry time of a subscription. These errors can cause the subscription to expire early (resulting in message CSQX456I being issued), or expire after the object has expired (resulting in erroneous MQRC 2085 (MQRC_UNKNOWN_OBJECT) errors).

For large clusters, it can be disruptive if many queue managers automatically resend all information about themselves at the same time. See Refreshing in a large cluster can affect performance and availability of the cluster.

Related concepts

"Clustering: Using REFRESH CLUSTER best practices" on page 66
You use the **REFRESH CLUSTER** command to discard all locally held information about a cluster and rebuild that information from the full repositories in the cluster. You should not need to use this command, except in exceptional circumstances. If you do need to use it, there are special considerations about how you use it. This information is a guide based on testing and feedback from customers.

Example clusters

The first example shows the smallest possible cluster of two queue managers. The second and third examples show two versions of a three queue manager cluster.

The smallest possible cluster contains only two queue managers. In this case both queue managers contain full repositories. You need only a few definitions to set up the cluster, and yet there is a high degree of autonomy at each queue manager.

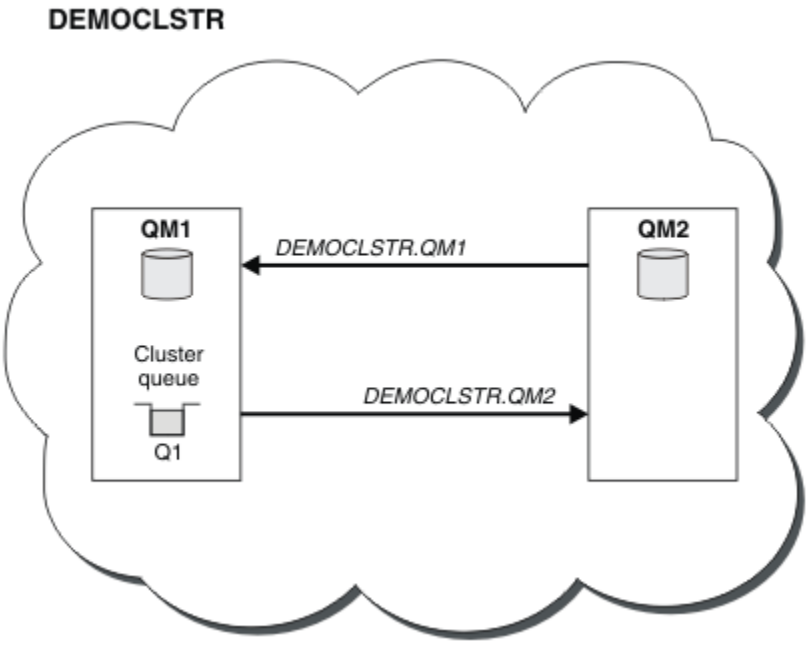


Figure 8. A small cluster of two queue managers

- The queue managers can have long names such as LONDON and NEWYORK. z/OS On IBM MQ for z/OS, queue manager names are limited to four characters.
- Each queue manager is typically configured on a separate machine. However, you can have multiple queue managers on the same machine.

For instructions on setting up a similar example cluster, see Setting up a new cluster.

Figure 9 on page 39 shows the components of a cluster called CLSTR1.

- In this cluster, there are three queue managers, QM1, QM2, and QM3.
- QM1 and QM2 host repositories of information about all the queue managers and cluster-related objects 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. Queues that are accessible to any other queue manager in the cluster are called *cluster queues*. The cluster queues are represented in the diagram by the shaded queues. Cluster queues are accessible from anywhere in the cluster. IBM MQ clustering code ensures that remote queue definitions for cluster queues are created on any queue manager that refers to them.

  As with distributed queuing, an application uses the MQPUT call to put a message on a cluster queue at any queue manager in the cluster. An application uses the MQGET call to retrieve messages from a cluster queue only on the queue manager where the queue resides.
- Each queue manager has a manually created definition for the receiving end of a channel called *cluster_name. queue_manager_name* on which it can receive messages. On the receiving queue manager, *cluster_name. queue_manager_name* is a cluster-receiver channel. A cluster-receiver channel is like a receiver channel used in distributed queuing; it receives messages for the queue manager. In addition, it also receives information about the cluster.

•

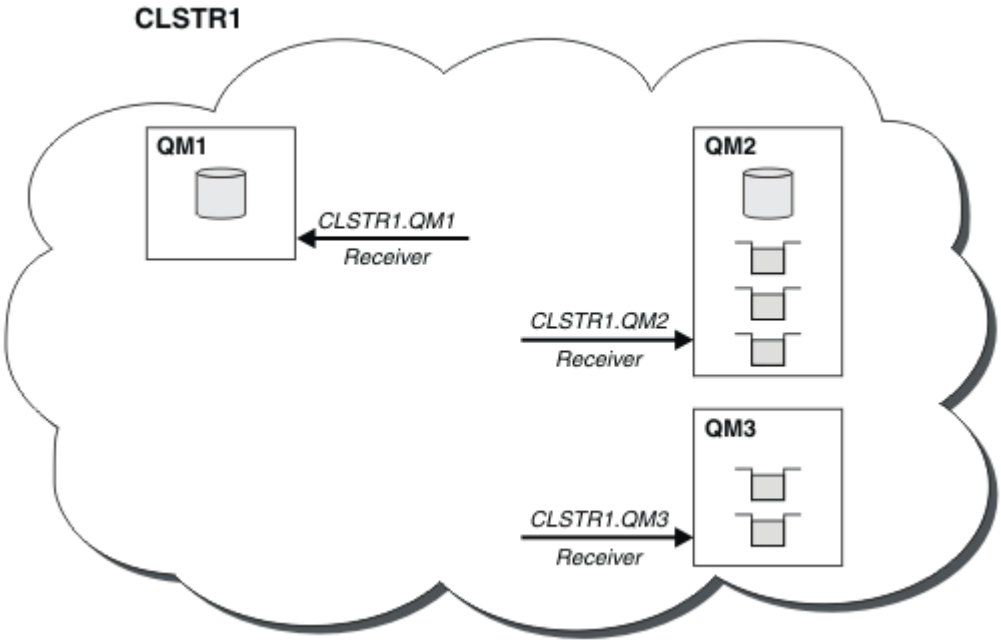


Figure 9. A cluster of queue managers

- In Figure 10 on page 39 each queue manager also has a definition for the sending end of a channel. It connects to the cluster-receiver channel of one of the full repository queue managers. On the sending queue manager, *cluster_name. queue_manager_name* is a cluster-sender channel. QM1 and QM3 have cluster-sender channels connecting to CLSTR1.QM2, see dotted line "2".

  QM2 has a cluster-sender channel connecting to CLSTR1.QM1, see dotted line "3". A cluster-sender channel is like a sender-channel used in distributed queuing; it sends messages to the receiving queue manager. In addition, it also sends information about the cluster.

  Once both the cluster-receiver end and the cluster-sender end of a channel are defined, the channel starts automatically.

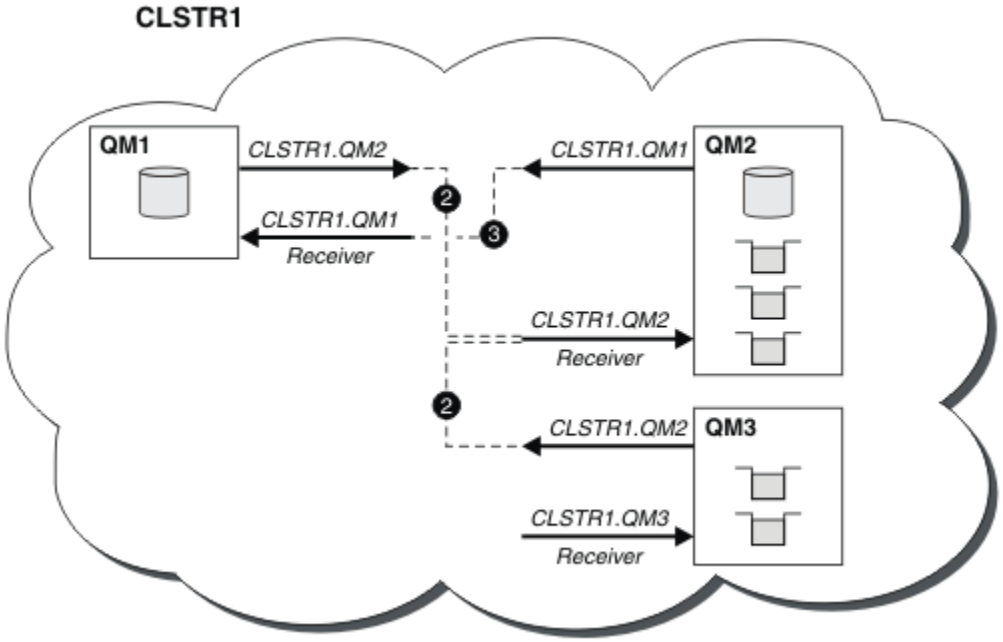


Figure 10. A cluster of queue managers with sender channels

Defining a cluster-sender channel on the local queue manager introduces that queue manager to one of the full repository queue managers. The full repository queue manager updates the information in its full repository accordingly. Then it automatically creates a cluster-sender channel back to the original queue manager, and sends that queue manager information about the cluster. Thus a queue manager learns about a cluster and a cluster learns about a queue manager.

Look again at Figure 9 on page 39. Suppose that an application connected to queue manager QM3 wants to send some messages to the queues at QM2. The first time that QM3 must access those queues, it discovers them by consulting a full repository. The full repository in this case is QM2, which is accessed using the sender channel CLSTR1.QM2. With the information from the repository, it can automatically create remote definitions for those queues. If the queues are on QM1, this mechanism still works, because QM2 is a full repository. A full repository has a complete record of all the objects in the cluster. In this latter case, QM3 would also automatically create a cluster-sender channel corresponding to the cluster-receiver channel on QM1, allowing direct communication between the two.

Figure 11 on page 40 shows the same cluster, with the two cluster-sender channels that were created automatically. The cluster-sender channels are represented by the two dashed lines that join with the cluster-receiver channel CLSTR1.QM3. It also shows the cluster transmission queue, SYSTEM.CLUSTER.TRANSMIT.QUEUE, which QM1 uses to send its messages. All queue managers in the cluster have a cluster transmission queue, from which they can send messages to any other queue manager in the same cluster.

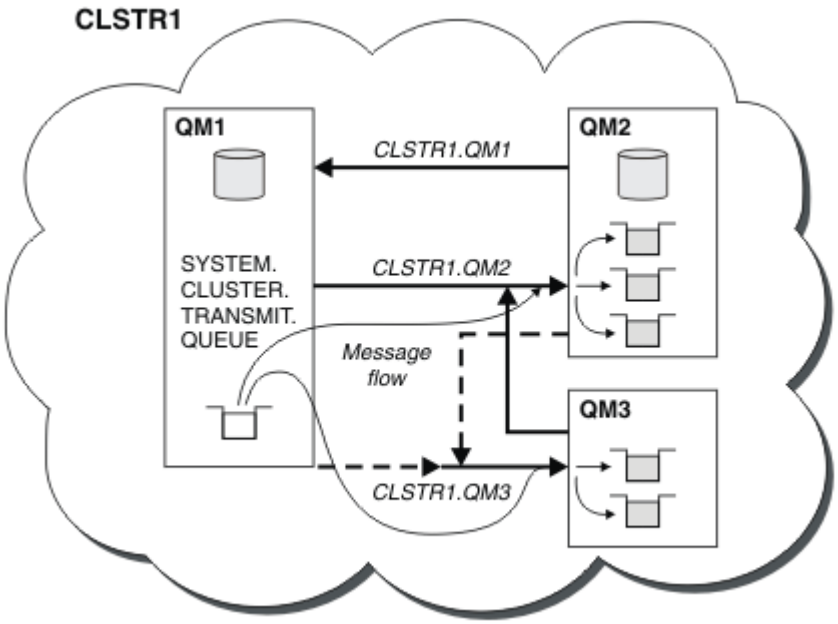


Figure 11. A cluster of queue managers, showing auto-defined channels

Note: Other diagrams show only the receiving ends of channels for which you make manual definitions. The sending ends are omitted because they are mostly defined automatically when needed. The auto-definition of most cluster-sender channels is crucial to the function and efficiency of clusters.

Related concepts

"Comparison of clustering and distributed queuing" on page 29
Compare the components that need to be defined to connect queue managers using distributed queuing and clustering.

Components of a cluster

Related tasks

Configuring a queue manager cluster

Setting up a new cluster

Clustering: Best practices

Clusters provide a mechanism for interconnecting queue managers. The best practices described in this section are based on testing and feedback from customers.

A successful cluster setup is dependent on good planning and a thorough understanding of IBM MQ fundamentals, such as good application management and network design. Ensure that you are familiar with the information in the related topics before continuing.

Related concepts

Distributed queuing and clusters

Clusters

Related tasks

"Designing clusters" on page 23

Clusters provide a mechanism for interconnecting queue managers in a way that simplifies both the initial configuration and the ongoing management. Clusters must be carefully designed, to ensure that they function correctly, and that they achieve the required levels of availability and responsiveness.

Monitoring clusters

Clustering: Special considerations for overlapping clusters

This topic provides guidance for planning and administering IBM MQ clusters. This information is a guide based on testing and feedback from customers.

Cluster ownership

Familiarize yourself with overlapping clusters before reading the following information. See "Overlapping clusters" on page 35 and Configuring message paths between clusters for the necessary information.

When configuring and managing a system that consists of overlapping clusters, it is best to adhere to the following:

- Although IBM MQ clusters are 'loosely coupled' as previously described, it is useful to consider a cluster as a single unit of administration. This concept is used because the interaction between definitions on individual queue managers is critical to the smooth functioning of the cluster. For example: When using workload balanced cluster queues it is important that a single administrator or team understand the full set of possible destinations for messages, which depends on definitions spread throughout the cluster. More trivially, cluster sender/receiver channel pairs must be compatible throughout.
- Considering this previous concept; where multiple clusters meet (which are to be administered by separate teams / individuals), it is important to have clear policies in place controlling administration of the gateway queue managers.
- It is useful to treat overlapping clusters as a single namespace: Channel names and queue manager names must be unique throughout a single cluster. Administration is much easier when unique throughout the entire topology. It is best to follow a suitable naming convention, possible conventions are described in "Cluster naming conventions" on page 33.
- Sometimes administrative and system management cooperation is essential. For example, cooperation between organizations that own different clusters that need to overlap. A clear understanding of who owns what, and enforceable rules and conventions helps clustering run smoothly when overlapping clusters.

Overlapping clusters: Gateways

In general, a single cluster is easier to administer than multiple clusters. Therefore creating large numbers of small clusters (one for every application for example) is something to be avoided generally.

However, to provide classes of service, you can implement overlapping clusters. For example:

- If you have concentric clusters where the smaller one is for Publish/Subscribe. See How to size systems for more information.

- If some queue managers are to be administered by different teams. See the previous section "Cluster ownership" on page 41for more information.
- If it makes sense from an organizational or geographical point of view.
- If equivalent clusters work with name resolution, for example when implementing TLS in an existing cluster.

There is no security benefit from overlapping clusters; allowing clusters administered by two different teams to overlap, effectively joins the teams as well as the topology:

- Any name advertised in such a cluster is accessible to the other cluster.
- Any name advertised in one cluster can be advertised in the other to draw off eligible messages.
- Any non-advertised object on a queue manager adjacent to the gateway can be resolved from any clusters of which the gateway is a member.

The namespace is the union of both clusters and must be treated as a single namespace. Therefore, ownership of an overlapping cluster is shared amongst all the administrators of both clusters.

When a system contains multiple clusters, there might be a requirement to route messages from queue managers in one cluster to queues on queue managers in another cluster. In this situation, the multiple clusters must be interconnected in some way: A good pattern to follow is the use of gateway queue managers between clusters. This arrangement avoids building up a difficult-to-manage mesh of point-to-point channels, and provides a good place to manage such issues as security policies. There are two distinct ways of achieving this arrangement:

1. Place one (or more) queue managers in both clusters using a second cluster receiver definition. This arrangement involves fewer administrative definitions but, as previously stated, means that ownership of an overlapping cluster is shared amongst all the administrators of both clusters.
2. Pair a queue manager in cluster one with a queue manager in cluster two using traditional point-to-point channels.

In either of these cases, various tools can be used to route traffic appropriately. In particular, queue or queue manager aliases can be used to route into the other cluster, and a queue manager alias with blank **RQMNAME** property re-drives workload balancing where it is wanted.

Related concepts

"Cluster naming conventions" on page 33
Consider naming queue managers in the same cluster using a naming convention that identifies the cluster to which the queue manager belongs. Use a similar naming convention for channel names and extend it to describe the channel characteristics.

Clustering: Topology design considerations
This topic provides guidance for planning and administering IBM MQ clusters. This information is a guide based on testing and feedback from customers.

By thinking about where user applications and internal administrative processes are going to be located in advance, many problems can either be avoided, or minimized at a later date. This topic contains information about design decisions that can improve performance, and simplify maintenance tasks as the cluster scales.

- "Performance of the clustering infrastructure" on page 42
- "Full repositories" on page 43
- "Should applications use queues on full repositories?" on page 44
- "Managing channel definitions" on page 45
- "Workload balancing over multiple channels" on page 45

Performance of the clustering infrastructure

When an application tries to open a queue on a queue manager in a cluster, the queue manager registers its interest with the full repositories for that queue so that it can learn where the queue exists in the

cluster. Any updates to the queue location or configuration are automatically sent by the full repositories to the interested queue manager. This registering of interest is internally known as a subscription (these subscriptions are not the same as IBM MQ subscriptions used for publish/subscribe messaging in IBM MQ)

All information about a cluster goes through every full repository. Full repositories are therefore always being used in a cluster for administrative message traffic. The high usage of system resources when managing these subscriptions, and the transmission of them and the resulting configuration messages, can cause a considerable load on the clustering infrastructure. There are a number of things to consider when ensuring that this load is understood and minimized wherever possible:

- The more individual queue managers using a cluster queue, the more subscriptions are in the system, and thus the bigger the administrative overhead when changes occur and interested subscribers need to be notified, especially on the full repository queue managers. One way to minimize unnecessary traffic and full repository load is by connecting similar applications (that is, those applications that work with the same queues) to a smaller number of queue managers.
- In addition to the number of subscriptions in the system affecting the performance the rate of change in the configuration of clustered objects can affect performance, for example the frequent changing of a clustered queue configuration.
- When a queue manager is a member of multiple clusters (that is, it is part of an overlapping cluster system) any interest made in a queue results in a subscription for each cluster it is a member of, even if the same queue managers are the full repositories for more than one of the clusters. This arrangement increases the load on the system, and is one reason to consider whether multiple overlapping clusters are necessary, rather than a single cluster.
- Application message traffic (that is, the messages being sent by IBM MQ applications to the cluster queues) does not go via the full repositories to reach the destination queue managers. This message traffic is sent directly between the queue manager where the message enters the cluster, and the queue manager where the cluster queue exists. It is not therefore necessary to accommodate high rates of application message traffic with respect to the full repository queue managers, unless the full repository queue managers happen to be either of those two queue managers mentioned. For that reason, it is recommended that full repository queue managers are not used for application message traffic in clusters where the clustering infrastructure load is significant.

Full repositories

A repository is a collection of information about the queue managers that are members of a cluster. A queue manager that hosts a complete set of information about every queue manager in the cluster has a full repository. For more information about full repositories and partial repositories, see Cluster repository.

Full repositories must be held on servers that are reliable and as highly available as possible and single points of failure must be avoided. The cluster design must always have two full repositories. If there is a failure of a full repository, the cluster can still operate.

Details of any updates to cluster resources made by a queue manager in a cluster; for example, clustered queues, are sent from that queue manager to two full repositories at most in that cluster (or to one if there is only one full repository queue manager in the cluster). Those full repositories hold the information and propagate it to any queue managers in the cluster that show an interest in it (that is, they subscribe to it). To ensure that each member of the cluster has an up-to-date view of the cluster resources there, each queue manager must be able to communicate with at least one full repository queue manager at any one time.

If, for any reason a queue manager cannot communicate with any full repositories, it can continue to function in the cluster based on its already cached level of information for a period time, but no new updates or access to previously unused cluster resources are available.

For this reason, you must aim to keep the two full repositories available at all times. However, this arrangement does not mean that extreme measures must be taken because the cluster functions adequately for a short while without a full repository.

There is another reason that a cluster must have two full repository queue managers, other than the availability of cluster information: This reason is to ensure that the cluster information held in the full repository cache exists in two places for recovery purposes. If there is only one full repository, and it loses its information about the cluster, then manual intervention on all queue managers within the cluster is required in order to get the cluster working again. If there are two full repositories however, then because information is always published to and subscribed for from two full repositories, the failed full repository can be recovered with the minimum of effort.

- It is possible to perform maintenance on full repository queue managers in a two full repository cluster design without impacting users of that cluster: The cluster continues to function with only one repository, so where possible bring the repositories down, apply the maintenance, and back up again one at a time. Even if there is an outage on the second full repository, running applications are unaffected for a minimum of three days.
- Unless there is a good reason for using a third repository, such as using a geographically local full repository for geographical reasons, use the two repository design. Having three full repositories means that you never know which are the two that are currently in use, and there might be administrative problems caused by interactions between multiple workload management parameters. It is not recommend to have more than two full repositories.
- If you still need better availability, consider hosting the full repository queue managers as multi-instance queue managers or using platform specific high availability support to improve their availability.
- You must fully interconnect all the full repository queue managers with manually defined cluster sender channels. Particular care must be taken when the cluster does have, for some justifiable reason, more than two full repositories. In this situation it is often possible to miss one or more channels and for it not to be immediately apparent. When full interconnection does not occur, hard to diagnose problems often arise. They are hard to diagnose because some full repositories not holding all repository data and therefore resulting in queue managers in the cluster having different views of the cluster depending on the full repositories that they connect to.

Should applications use queues on full repositories?

A full repository is in most ways exactly like any other queue manager, and it is therefore possible to host application queues on the full repository and connect applications directly to these queue managers. Should applications use queues on full repositories?

The commonly accepted answer is "No?". Although this configuration is possible, many customers prefer to keep these queue managers dedicated to maintaining the full repository cluster cache. Points to consider when deciding on either option are described here, but ultimately the cluster architecture must be appropriate to the particular demands of the environment.

- Upgrades: Usually, in order to use new cluster features in new releases of IBM MQ the full repository queue managers of that cluster must be upgraded first. When an application in the cluster wants to use new features, it might be useful to be able to update the full repositories (and some subset of partial repositories) without testing a number of co-located applications.
- Maintenance: In a similar way if you must apply urgent maintenance to the full repositories, they can be restarted or refreshed with the **REFRESH** command without touching applications.
- Performance: As clusters grow and demands on the full repository cluster cache maintenance become greater, keeping applications separate reduces risk of this affecting application performance through contention for system resources.
- Hardware requirements: Typically, full repositories do not need to be powerful; for example, a simple UNIX server with a good expectation of availability is sufficient. Alternatively, for very large or constantly changing clusters, the performance of the full repository computer must be considered.
- Software requirements: Requirements are usually the main reason for choosing to host application queues on a full repository. In a small cluster, collocation might mean a requirement for fewer queue managers/servers over all.

Managing channel definitions

Even within a single cluster, multiple channel definitions can exist giving multiple routes between two queue managers.

There is sometimes an advantage to having parallel channels within a single cluster, but this design decision must be considered thoroughly; apart from adding complexity, this design might result in channels being under-used which reduces performance. This situation occurs because testing usually involves sending lots of messages at a constant rate, so the parallel channels are fully used. But with real-world conditions of a non-constant stream of messages, the workload balancing algorithm causes performance to drop as the message flow is switched from channel to channel.

When a queue manager is a member of multiple clusters, the option exists to use a single channel definition with a cluster namelist, rather than defining a separate CLUSRCVR channel for each cluster. However, this setup can cause administration difficulties later; consider for example the case where TLS is to be applied to one cluster but not a second. It is therefore preferable to create separate definitions, and the naming convention suggested in "Cluster naming conventions" on page 33 supports this.

Workload balancing over multiple channels

This information is intended as an advanced understanding of the subject. For the basic explanation of this subject (which must be understood before using the information here), see Using clusters for workload management, Workload balancing in clusters, and The cluster workload management algorithm.

The cluster workload management algorithm provides a large set of tools, but they must not all be used with each other without fully understanding how they work and interact. It might not be immediately obvious how important channels are to the workload balancing process: The workload management round-robin algorithm behaves as though multiple cluster channels to a queue manager that owns a clustered queue, are treated as multiple instances of that queue. This process is explained in more detail in the following example:

1. There are two queue managers hosting a queue in a cluster: QM1 and QM2.
2. There are five cluster receiver channels to QM1.
3. There is only one cluster receiver channel to QM2.
4. When **MQPUT** or **MQOPEN** on QM3 chooses an instance, the algorithm is five times more likely to send the message to QM1 than to QM2.
5. The situation in step 4 occurs because the algorithm sees six options to choose from (5+1) and round-robins across all five channels to QM1 and the single channel to QM2.

Another subtle behavior is that even when putting messages to a clustered queue that happens to have one instance configured on the local queue manager, IBM MQ uses the state of the local cluster receiver channel to decide whether messages are to be put to the local instance of the queue or remote instances of the queue. In this scenario:

1. When putting messages the workload management algorithm does not look at individual cluster queues, it looks at the cluster channels which can reach those destinations.
2. To reach local destinations, the local receiver channels are included in this list (although they are not used to send the message).
3. When a local receiver channel is stopped, the workload management algorithm, prefers an alternative instance by default if its CLUSRCVR is not stopped. If there are multiple local CLUSRCVR instances for the destination and at least one is not stopped, the local instance remains eligible.

Clustering: Application isolation using multiple cluster transmission queues
You can isolate the message flows between queue managers in a cluster. You can place messages being transported by different cluster-sender channels onto different cluster transmission queues. You can use the approach in a single cluster or with overlapping clusters. The topic provides examples and some best practices to guide you in choosing an approach to use.

When you deploy an application, you have a choice over which IBM MQ resources it shares with other applications and which resources it does not share. There are a number of types of resources that can

be shared, the main ones being the server itself, the queue manager, channels, and queues. You can choose to configure applications with fewer shared resources; allocating separate queues, channels, queue managers, or even servers to individual applications. If you do so, the overall system configuration becomes bigger and more complex. Using IBM MQ clusters reduces the complexity of managing more servers, queue managers, queues, and channels, but it introduces another shared resource, the cluster transmission queue, SYSTEM.CLUSTER.TRANSMIT.QUEUE.

Figure 12 on page 47 is a slice out of a large IBM MQ deployment that illustrates the significance of sharing SYSTEM.CLUSTER.TRANSMIT.QUEUE. In the diagram, the application, Client App, is connected to the queue manager QM2 in cluster CL1. A message from Client App is processed by the application, Server App. The message is retrieved by Server App from the cluster queue Q1 on the queue manager QM3 in CLUSTER2. Because the client and server applications are not in the same cluster, the message is transferred by the gateway queue manager QM1.

The normal way to configure a cluster gateway is to make the gateway queue manager a member of all the clusters. On the gateway queue manager are defined clustered alias queues for cluster queues in all the clusters. The clustered queue aliases are available in all the clusters. Messages put to the cluster queue aliases are routed via the gateway queue manager to their correct destination. The gateway queue manager puts messages sent to the clustered alias queues onto the common SYSTEM.CLUSTER.TRANSMIT.QUEUE on QM1.

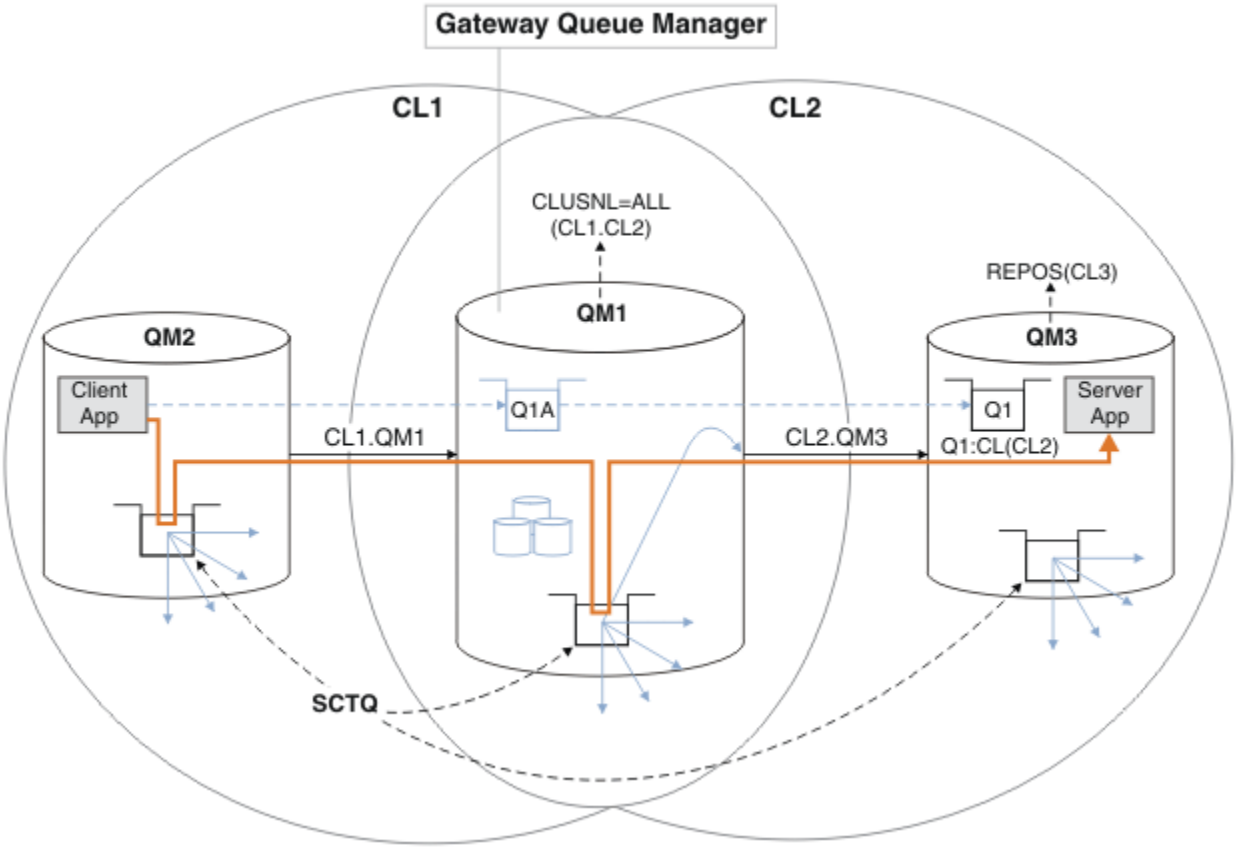
The hub and spoke architecture requires all messages between clusters to pass through the gateway queue manager. The result is that all messages flow through the single cluster transmission queue on QM1, SYSTEM.CLUSTER.TRANSMIT.QUEUE.

From a performance perspective, a single queue is not a problem. A common transmission queue generally does not represent a performance bottleneck. Message throughput on the gateway is largely determined by the performance of the channels that connect to it. Throughput is not generally affected by the number of queues, or the number of messages on the queues that use the channels.

From some other perspectives, using a single transmission queue for multiple applications has drawbacks:

- You cannot isolate the flow of messages to one destination from the flow of messages to another destination. You cannot separate the storage of messages before they are forwarded, even if the destinations are in different clusters on different queue managers.

  If one cluster destination becomes unavailable, messages for that destination build-up in the single transmission queue, and eventually the messages fill it up. Once the transmission queue is full, it stops messages from being placed onto the transmission queue for any cluster destination.
- It is not easy to monitor the transfer of messages to different cluster destinations. All the messages are on the single transmission queue. Displaying the depth of the transmission queue gives you little indication whether messages are being transferred to all destinations.



Note: The arrows in Figure 12 on page 47 and following figures are of different types. Solid arrows represent message flows. The labels on solid arrows are message channel names. The gray solid arrows are potential message flows from the SYSTEM.CLUSTER.TRANSMIT.QUEUE onto cluster-sender channels. Black dashed lines connect labels to their targets. Gray dashed arrows are references; for example from an MQOPEN call by Client App to the cluster alias queue definition Q1A.

Figure 12. Client-server application deployed to hub and spoke architecture using IBM MQ clusters

In Figure 12 on page 47, clients of Server App open the queue Q1A. Messages are put to SYSTEM.CLUSTER.TRANSMIT.QUEUE on QM2, transferred to SYSTEM.CLUSTER.TRANSMIT.QUEUE on QM1, and then transferred to Q1 on QM3, where they are received by the Server App application.

The message from Client App passes through system cluster transmission queues on QM2 and QM1. In Figure 12 on page 47, the objective is to isolate the message flow on the gateway queue manager from the client application, so that its messages are not stored on SYSTEM.CLUSTER.TRANSMIT.QUEUE. You can isolate flows on any of the other clustered queue managers. You can also isolate flows in the other direction, back to the client. To keep the descriptions of the solutions brief, the descriptions consider only a single flow from the client application.

Solutions to isolating cluster message traffic on a cluster gateway queue manager

One way to solve the problem is to use queue manager aliases, or remote queue definitions, to bridge between clusters. Create a clustered remote queue definition, a transmission queue, and a channel, to separate each message flow on the gateway queue manager; see Adding a remote queue definition to isolate messages sent from a gateway queue manager.

From IBM WebSphere® MQ 7.5 onwards, cluster queue managers are not limited to a single cluster transmission queue. You have two choices:

1. Define additional cluster transmission queues manually, and define which cluster-sender channels transfer messages from each transmission queue; see Adding a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager.

2. Allow the queue manager to create and manage additional cluster transmission queues automatically. It defines a different cluster transmission queue for each cluster-sender channel; see Changing the default to separate cluster transmission queues to isolate message traffic.

You can combine manually defined cluster transmission queues for some cluster-sender channels, with the queue manager managing the rest. The combination of transmission queues is the approach taken in Adding a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager. In that solution, most messages between clusters use the common SYSTEM.CLUSTER.TRANSMIT.QUEUE. One application is critical, and all its message flows are isolated from other flows by using one manually defined cluster transmission queue.

The configuration in Adding a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager is limited. It does not separate the message traffic going to a cluster queue on the same queue manager in the same cluster as another cluster queue. You can separate the message traffic to individual queues by using the remote queue definitions that are part of distributed queuing. With clusters, using multiple cluster transmission queues, you can separate message traffic that goes to different cluster-sender channels. Multiple cluster queues in the same cluster, on the same queue manager, share a cluster-sender channel. Messages for those queues are stored on the same transmission queue, before being forwarded from the gateway queue manager. In the configuration in Adding a cluster and a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager, the limitation is side-stepped by adding another cluster and making the queue manager and cluster queue a member of the new cluster. The new queue manager might be the only queue manager in the cluster. You could add more queue managers to the cluster, and use the same cluster to isolate cluster queues on those queue managers as well.

Related concepts

"Access control and multiple cluster transmission queues" on page 28
Choose between three modes of checking when an application puts messages to remote cluster queues. The modes are checking remotely against the cluster queue, checking locally against SYSTEM.CLUSTER.TRANSMIT.QUEUE, or checking against local profiles for the cluster queue, or cluster queue manager.

Working with cluster transmission queues and cluster-sender channels

"Overlapping clusters" on page 35
Overlapping clusters provide additional administrative capabilities. Use namelists to reduce the number of commands needed to administer overlapping clusters.

Related tasks

Authorizing putting messages on remote cluster queues

Adding a remote queue definition to isolate messages sent from a gateway queue manager

Adding a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager

Adding a cluster and a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager

Changing the default to separate cluster transmission queues to isolate message traffic

Creating two-overlapping clusters with a gateway queue manager

Configuring message paths between clusters

Securing

Related reference

setmqaut

Clustering: Planning how to configure cluster transmission queues
You are guided through the choices of cluster transmission queues. You can configure one common default queue, separate default queues, or manually defined queues.

Before you begin

Review "How to choose what type of cluster transmission queue to use" on page 51.

About this task

You have some choices to make when you are planning how to configure a queue manager to select a cluster transmission queue.

1. What is the default cluster transmission queue for cluster message transfers?
   a. A common cluster transmission queue, `SYSTEM.CLUSTER.TRANSMIT.QUEUE`.
   b. Separate cluster transmission queues. The queue manager manages the separate cluster transmission queues. It creates them as permanent-dynamic queues from the model queue, `SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE`. It creates one cluster transmission queue for each cluster-sender channel it uses.
2. For the cluster transmission queues that you do decide to create manually, you have a further two choices:
   a. Define a separate transmission queue for each cluster-sender channel that you decide to configure manually. In this case, set the **CLCHNAME** queue attribute of the transmission queue to the name of a cluster-sender channel. Select the cluster-sender channel that is to transfer messages from this transmission queue.
   b. Combine message traffic for a group of cluster-sender channels onto the same cluster transmission queue; see Figure 13 on page 50. In this case, set the **CLCHNAME** queue attribute of each common transmission queue to a generic cluster-sender channel name. A generic cluster-sender channel name is a filter to group cluster-sender channel names. For example, `SALES.*` groups all cluster-sender channels that have names beginning with `SALES.`. You can place multiple wildcard characters anywhere in the filter-string. The wildcard character is an asterisk, "`*`". It represents from zero to any number of characters.

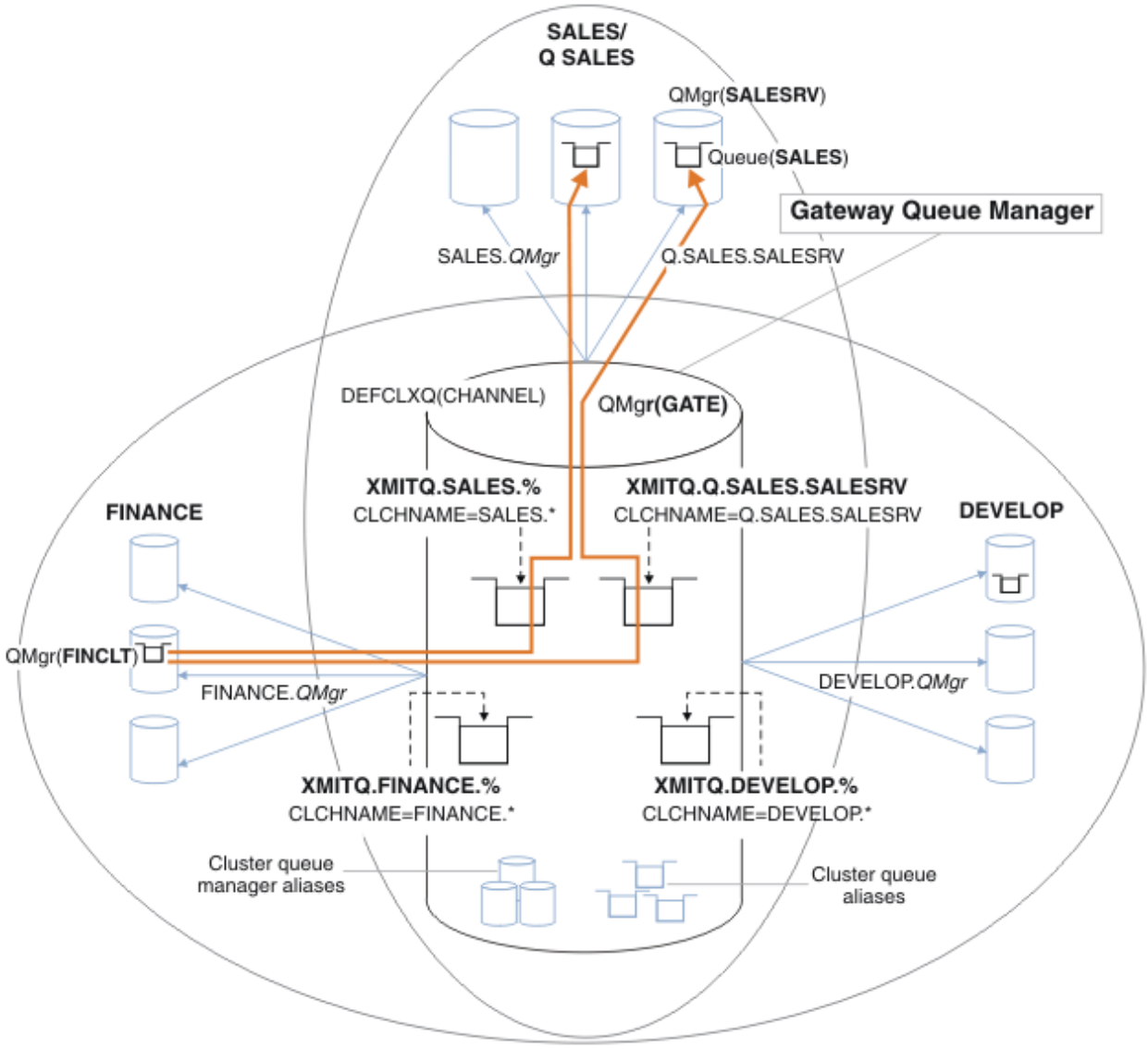


Figure 13. Example of specific transmission queues for different departmental IBM MQ clusters

Procedure

1. Select the type of default cluster transmission queue to use.
   - Choose a single cluster transmission queue, or separate queues for each cluster connection.

   Leave the default setting or run the **MQSC** command:

   ```
   ALTER QMGR DEFCLXQ(CHANNEL)
   ```

2. Isolate any message flows that must not share a cluster transmission queue with other flows.
   - See "Clustering: Example configuration of multiple cluster transmission queues" on page 53. In the example the SALES queue, which must be isolated, is a member of the SALES cluster, on SALESRV. To isolate the SALES queue, create a new cluster `Q.SALES`, make the SALESRV queue manager a member, and modify the SALES queue to belong to `Q.SALES`.
   - Queue managers that send messages to SALES must also be members of the new cluster. If you use a clustered queue alias and a gateway queue manager, as in the example, in many cases you can limit the changes to making the gateway queue manager a member of the new cluster.
   - However, separating flows from the gateway to the destination does not separate flows to the gateway from the source queue manager. But it sometimes turns out to be sufficient to separate

flows from the gateway and not flows to the gateway. If it is not sufficient, then add the source queue manager into the new cluster. If you want messages to travel through the gateway, move the cluster alias to the new cluster and continue to send messages to the cluster alias on the gateway, and not directly to the target queue manager.

Follow these steps to isolate message flows:

a) Configure the destinations of the flows so that each target queue is the only queue in a specific cluster, on that queue manager.

b) Create the cluster-sender and cluster-receiver channels for any new clusters you have created following a systematic naming convention.

- See "Clustering: Special considerations for overlapping clusters" on page 41.

c) Define a cluster transmission queue for each isolated destination on every queue manager that sends messages to the target queue.

- A naming convention for cluster transmission queues is to use the value of the cluster channel name attribute, `CLCHNAME`, prefixed with `XMITQ.`

3. Create cluster transmission queues to meet governance or monitoring requirements.

- Typical governance and monitoring requirements result in a transmission queue per cluster or a transmission queue per queue manager. If you follow the naming convention for cluster channels, *`ClusterName. QueueManagerName`*, it is easy to create generic channel names that select a cluster of queue managers, or all the clusters a queue manager is a member of; see "Clustering: Example configuration of multiple cluster transmission queues" on page 53.
- Extend the naming convention for cluster transmission queues to cater for generic channel names, by replacing the asterisk symbol by a percent sign. For example,

```
DEFINE QLOCAL(XMITQ.SALES.%) USAGE(XMITQ) CLCHNAME(SALES.*)
```

Related concepts

Working with cluster transmission queues and cluster-sender channels

"Access control and multiple cluster transmission queues" on page 28

Choose between three modes of checking when an application puts messages to remote cluster queues. The modes are checking remotely against the cluster queue, checking locally against `SYSTEM.CLUSTER.TRANSMIT.QUEUE`, or checking against local profiles for the cluster queue, or cluster queue manager.

"Overlapping clusters" on page 35

Overlapping clusters provide additional administrative capabilities. Use namelists to reduce the number of commands needed to administer overlapping clusters.

Related tasks

Adding a remote queue definition to isolate messages sent from a gateway queue manager

Adding a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager

Adding a cluster and a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager

Changing the default to separate cluster transmission queues to isolate message traffic

Creating two-overlapping clusters with a gateway queue manager

Configuring message paths between clusters

How to choose what type of cluster transmission queue to use

How to choose between different cluster transmission queue configuration options.

You can choose which cluster transmission queue is associated with a cluster-sender channel.

1. You can have all cluster-sender channels associated with the single default cluster transmit queue, `SYSTEM.CLUSTER.TRANSMIT.QUEUE`; this option is the default.
2. You can set all cluster-sender channels to be automatically associated with a separate cluster transmission queue. The queues are created by the queue manager from the model

queue SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE and named SYSTEM.CLUSTER.TRANSMIT.*ChannelName*. Channels will use their uniquely named cluster transmit queue if the queue manager attribute **DEFCLXQ** is set to CHANNEL.

3. You can set specific cluster-sender channels to be served by a single cluster transmission queue. Select this option by creating a transmission queue and setting its **CLCHNAME** attribute to the name of the cluster-sender channel.
4. You can select groups of cluster-sender channels to be served by a single cluster transmission queue. Select this option by creating a transmission queue and setting its **CLCHNAME** attribute to a generic channel name, such as *ClusterName*.*. If you name cluster channels by following the naming conventions in "Clustering: Special considerations for overlapping clusters" on page 41, this name selects all cluster channels connected to queue managers in the cluster *ClusterName*.

You can combine either of the default cluster transmission queue options for some cluster-sender channels, with any number of specific and generic cluster transmission queue configurations.

Best practices

In most cases, for existing IBM MQ installations, the default configuration is the best choice. A cluster queue manager stores cluster messages on a single cluster transmission queue, SYSTEM.CLUSTER.TRANSMIT.QUEUE. You have the choice of changing the default to storing messages for different queue managers and different clusters on separate transmission queues, or of defining your own transmission queues.

In most cases, for new IBM MQ installations, the default configuration is also the best choice. The process of switching from the default configuration to the alternative default of having one transmission queue for each cluster-sender channel is automatic. Switching back is also automatic. The choice of one or the other is not critical, you can reverse it.

The reason for choosing a different configuration is more to do with governance, and management, than with functionality or performance. With a couple of exceptions, configuring multiple cluster transmission queues does not benefit the behavior of the queue manager. It results in more queues, and requires you to modify monitoring and management procedures you have already set up that refer to the single transmission queue. That is why, on balance, remaining with the default configuration is the best choice, unless you have strong governance or management reasons for a different choice.

The exceptions are both concerned with what happens if the number of messages stored on SYSTEM.CLUSTER.TRANSMIT.QUEUE increases. If you take every step to separate the messages for one destination from the messages for another destination, then channel and delivery problems with one destination ought not to affect the delivery to another destination. However, the number of messages stored on SYSTEM.CLUSTER.TRANSMIT.QUEUE can increase due to not delivering messages fast enough to one destination. The number of messages on SYSTEM.CLUSTER.TRANSMIT.QUEUE for one destination can affect the delivery of messages to other destinations.

To avoid problems that result from filling up a single transmission queue, aim to build sufficient capacity into your configuration. Then, if a destination fails and a message backlog starts to build up, you have time to fix the problem.

If messages are routed through a hub queue manager, such as a cluster gateway, they share a common transmission queue, SYSTEM.CLUSTER.TRANSMIT.QUEUE. If the number of messages stored on SYSTEM.CLUSTER.TRANSMIT.QUEUE on the gateway queue manager reaches its maximum depth, the queue manager starts to reject new messages for the transmission queue until its depth reduces. The congestion affects messages for all destinations that are routed through the gateway. Messages back up the transmission queues of other queue managers that are sending messages to the gateway. The problem manifests itself in messages written to queue manager error logs, falling message throughput, and longer elapsed times between sending a message and the time that a message arrives at its destination.

The effect of congestion on a single transmission queue can become apparent, even before it is full. If you have a mixed message traffic, with some large non-persistent messages and some small messages, the time to deliver small messages increases as the transmission queue fills. The delay is due to writing

large non-persistent messages to disk that would not normally get written to disk. If you have time critical message flows, sharing a cluster transmission queue with other mixed messages flows, it could be worth configuring a special message path to isolate it from other message flows; see Adding a cluster and a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager.

The other reasons for configuring separate cluster transmission queues are to meet governance requirements, or to simplify monitoring messages that are sent to different cluster destinations. For example, you might have to demonstrate that messages for one destination never share a transmission queue with messages for another destination.

Change the queue manager attribute **DEFCLXQ** that controls the default cluster transmission queue, to create different cluster transmission queues for every cluster-sender channel. Multiple destinations can share a cluster-sender channel, so you must plan your clusters to meet this objective fully. Apply the method Adding a cluster and a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager systematically to all your cluster queues. The result you are aiming for is for no cluster destination to share a cluster-sender channel with another cluster destination. As a consequence, no message for a cluster destination shares its cluster transmission queue with a message for another destination.

Creating a separate cluster transmission queue for some specific message flow, makes it easy to monitor the flow of messages to that destination. To use a new cluster transmission queue, define the queue, associate it with a cluster-sender channel, and stop and start the channel. The change does not have to be permanent. You could isolate a message flow for a while, to monitor the transmission queue, and then revert to using the default transmission queue again.

Related tasks

Clustering: Example configuration of multiple cluster transmission queues
In this task you apply the steps to plan multiple cluster transmission queues to three overlapping clusters. The requirements are to separate messages flows to one cluster queue, from all other message flows, and to store messages for different clusters on different cluster transmission queues.

Clustering: Switching cluster transmission queues
Plan how the changes to the cluster transmission queues of an existing production queue manager are going to be brought into effect.

Clustering: Example configuration of multiple cluster transmission queues
In this task you apply the steps to plan multiple cluster transmission queues to three overlapping clusters. The requirements are to separate messages flows to one cluster queue, from all other message flows, and to store messages for different clusters on different cluster transmission queues.

About this task

The steps in this task show how to apply the procedure in “Clustering: Planning how to configure cluster transmission queues” on page 48 and arrive at the configuration shown in Figure 14 on page 54. It is an example of three overlapping clusters, with a gateway queue manager, that is configured with separate cluster transmission queues. The MQSC commands to define the clusters are described in “Creating the example clusters” on page 56.

For the example, there are two requirements. One is to separate the message flow from the gateway queue manager to the sales application that logs sales. The second is to query how many messages are waiting to be sent to different departmental areas at any point in time. The SALES, FINANCE, and DEVELOP clusters are already defined. Cluster messages are currently forwarded from SYSTEM.CLUSTER.TRANSMIT.QUEUE.

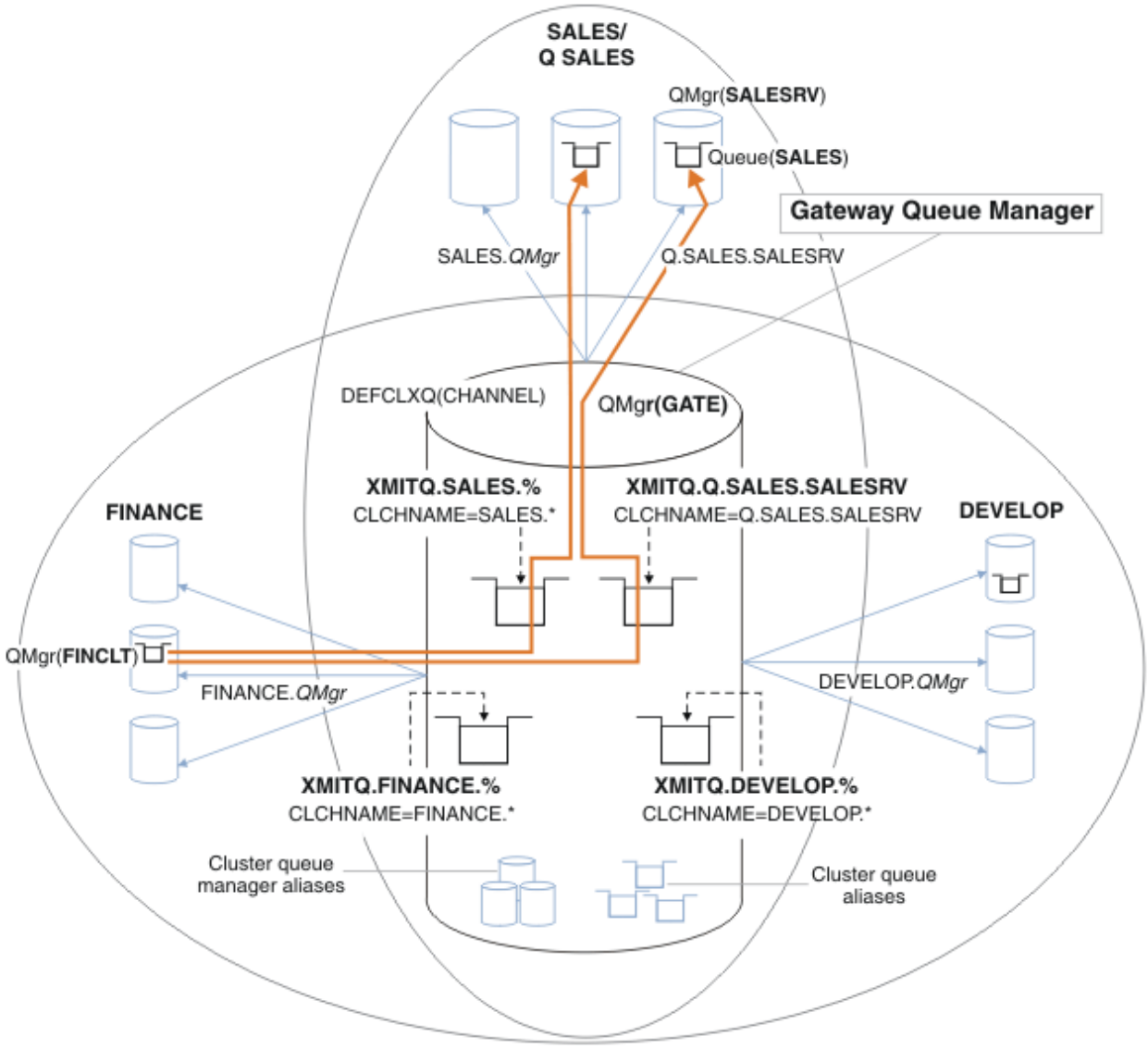


Figure 14. Example of specific transmission queues for different departmental IBM MQ clusters

The steps to modify the clusters are as follows. For the definitions, see Changes to isolate the sales queue in a new cluster and separate the gateway cluster transmission queues.

Procedure

1. The first configuration step is to " Select the type of default cluster transmission queue to use ".

   The decision is to create separate default cluster transmission queues by running the following **MQSC** command on the GATE queue manager.

   ```
   ALTER QMGR DEFCLXQ(CHANNEL)
   ```

   There is no strong reason for choosing this default, as the intention is to manually define cluster transmission queues. The choice does have a weak diagnostic value. If a manual definition is done wrongly, and a message flows down a default cluster transmission queue, it shows up in the creation of a permanent-dynamic cluster transmission queue.
2. The second configuration step is to " Isolate any message flows that must not share a cluster transmission queue with other flows ".

In this case the sales application that receives messages from the queue SALES on SALESRV requires isolation. Only isolation of messages from the gateway queue manager is required. The three sub-steps are:

a) " Configure the destinations of the flows so that each target queue is the only queue in a specific cluster, on that queue manager ".

The example requires adding the queue manager SALESRV to a new cluster within the sales department. If you have few queues that require isolation, you might decide on creating a specific cluster for the SALES queue. A possible naming convention for the cluster name is to name such clusters, `Q.` *QueueName*, for example `Q.SALES`. An alternative approach, which might be more practical if you have a large number of queues to be isolated, is to create clusters of isolated queues where and when needed. The clusters names might be `QUEUES.` *n*.

In the example, the new cluster is called `Q.SALES`. To add the new cluster, see the definitions in Changes to isolate the sales queue in a new cluster and separate the gateway cluster transmission queues. The summary of definition changes is as follows:

i) Add `Q.SALES` to the namelist of clusters on the repository queue managers. The namelist is referred to in the queue manager **REPOSNL** parameter.

ii) Add `Q.SALES` to the namelist of clusters on the gateway queue manager. The namelist is referred to in all the cluster queue alias and cluster queue manager alias definitions on the gateway queue manager.

iii) Create a namelist on the queue manager SALESRV, for both the clusters it is a member of, and change the cluster membership of the SALES queue:

```
DEFINE NAMELIST(CLUSTERS) NAMES(SALES, Q.SALES) REPLACE
ALTER QLOCAL(SALES) CLUSTER(' ') CLUSNL(SALESRV.CLUSTERS)
```

The SALES queue is a member of both clusters, just for the transition. Once the new configuration is running, you remove the SALES queue from the SALES cluster; see Figure 15 on page 59.

b) " Create the cluster-sender and cluster-receiver channels for any new clusters you have created following a systematic naming convention ".

i) Add the cluster-receiver channel `Q.SALES.` *RepositoryQMgr* to each of the repository queue managers

ii) Add the cluster-sender channel `Q.SALES.` *OtherRepositoryQMgr* to each of the repository queue managers, to connect to the other repository manager. Start these channels.

iii) Add the cluster receiver channels `Q.SALES.SALESRV`, and `Q.SALES.GATE` to either of the repository queue managers that is running.

iv) Add the cluster-sender channels `Q.SALES.SALESRV`, and `Q.SALES.GATE` to the SALESRV and GATE queue managers. Connect the cluster-sender channel to the repository queue manager that you created the cluster-receiver channels on.

c) " Define a cluster transmission queue for each isolated destination on every queue manager that sends messages to the target queue ".

On the gateway queue manager define the cluster transmission queue `XMITQ.Q.SALES.SALESRV` for the `Q.SALES.SALESRV` cluster-sender channel:

```
DEFINE QLOCAL(XMITQ.Q.SALES.SALESRV) USAGE(XMITQ) CLCHNAME(Q.SALES.SALESRV) REPLACE
```

3. The third configuration step is to " Create cluster transmission queues to meet governance or monitoring requirements ".

On the gateway queue manager define the cluster transmission queues:

```
DEFINE QLOCAL(XMITQ.SALES)  USAGE(XMITQ) CLCHNAME(SALES.*)  REPLACE
DEFINE QLOCAL(XMITQ.DEVELOP) USAGE(XMITQ) CLCHNAME(DEVELOP.*) REPLACE
DEFINE QLOCAL(XMITQ.FINANCE) USAGE(XMITQ) CLCHNAME(SALES.*)  REPLACE
```

What to do next

Switch to the new configuration on the gateway queue manager.

The switch is triggered by starting the new channels, and restarting the channels that are now associated with different transmission queues. Alternatively, you can stop and start the gateway queue manager.

1. Stop the following channels on the gateway queue manager:

```
SALES. Qmgr
DEVELOP. Qmgr
FINANCE. Qmgr
```

2. Start the following channels on the gateway queue manager:

```
SALES. Qmgr
DEVELOP. Qmgr
FINANCE. Qmgr
Q.SALES.SAVESRV
```

When the switch is complete, remove the SALES queue from the SALES cluster; see Figure 15 on page 59.

Related concepts

How to choose what type of cluster transmission queue to use
How to choose between different cluster transmission queue configuration options.

Related tasks

Clustering: Switching cluster transmission queues
Plan how the changes to the cluster transmission queues of an existing production queue manager are going to be brought into effect.

Creating the example clusters
The definitions and instructions to create the example cluster, and modify it to isolate the SALES queue and separate messages on the gateway queue manager.

About this task

The full **MQSC** commands to create the FINANCE, SALES, and Q.SALES clusters are provided in Definitions for the basic clusters, Changes to isolate the sales queue in a new cluster and separate the gateway cluster transmission queues, and Remove the sales queue on queue manager SALESRV from the sales cluster. The DEVELOP cluster is omitted from the definitions, to keep the definitions shorter.

Procedure

1. Create the SALES and FINANCE clusters, and the gateway queue manager.
   a) Create the queue managers.

   Run the command: `crtmqm -sax -u SYSTEM.DEAD.LETTER.QUEUE` *QmgrName* for each of the queue manager names in Table 4 on page 56.

Table 4. Queue manager names and port numbers

Description	Queue manager name	Port number
Finance repository	FINR1	1414
Finance repository	FINR2	1415
Finance client	FINCLT	1418
Sales repository	SALER1	1416
Sales repository	SALER2	1417

Table 4. Queue manager names and port numbers (continued)		
Description	**Queue manager name**	**Port number**
Sales server	SALESRV	1419
Gateway	GATE	1420

b) Start all the queue managers

Run the command: `strmqm` *QmgrName* for each of the queue manager names in Table 4 on page 56.

c) Create the definitions for each of the queue managers

Run the command: `runmqsc` *QmgrName* < *filename* where the files are listed in Definitions for the basic clusters, and the file name matches the queue manager name.

Definitions for the basic clusters

finr1.txt

```
DEFINE LISTENER(1414) TRPTYPE(TCP) IPADDR(localhost) CONTROL(QMGR) PORT(1414) REPLACE
START LISTENER(1414)
ALTER QMGR REPOS(FINANCE)
DEFINE CHANNEL(FINANCE.FINR2) CHLTYPE(CLUSSDR) CONNAME('localhost(1415)')
CLUSTER(FINANCE) REPLACE
DEFINE CHANNEL(FINANCE.FINR1) CHLTYPE(CLUSRCVR) CONNAME('localhost(1414)')
CLUSTER(FINANCE) REPLACE
```

finr2.txt

```
DEFINE LISTENER(1415) TRPTYPE(TCP) IPADDR(localhost) CONTROL(QMGR) PORT(1415) REPLACE
START LISTENER(1415)
ALTER QMGR REPOS(FINANCE)
DEFINE CHANNEL(FINANCE.FINR1) CHLTYPE(CLUSSDR) CONNAME('localhost(1414)')
CLUSTER(FINANCE) REPLACE
DEFINE CHANNEL(FINANCE.FINR2) CHLTYPE(CLUSRCVR) CONNAME('localhost(1415)')
CLUSTER(FINANCE) REPLACE
```

finclt.txt

```
DEFINE LISTENER(1418) TRPTYPE(TCP) IPADDR(localhost) CONTROL(QMGR) PORT(1418) REPLACE
START LISTENER(1418)
DEFINE CHANNEL(FINANCE.FINR1) CHLTYPE(CLUSSDR)  CONNAME('localhost(1414)')
CLUSTER(FINANCE) REPLACE
DEFINE CHANNEL(FINANCE.FINCLT) CHLTYPE(CLUSRCVR) CONNAME('localhost(1418)')
CLUSTER(FINANCE) REPLACE
DEFINE QMODEL(SYSTEM.SAMPLE.REPLY) REPLACE
```

saler1.txt

```
DEFINE LISTENER(1416) TRPTYPE(TCP) IPADDR(localhost) CONTROL(QMGR) PORT(1416) REPLACE
START LISTENER(1416)
ALTER QMGR REPOS(SALES)
DEFINE CHANNEL(SALES.SALER2) CHLTYPE(CLUSSDR) CONNAME('localhost(1417)')
CLUSTER(SALES) REPLACE
DEFINE CHANNEL(SALES.SALER1) CHLTYPE(CLUSRCVR) CONNAME('localhost(1416)')
CLUSTER(SALES) REPLACE
```

saler2.txt

```
DEFINE LISTENER(1417) TRPTYPE(TCP) IPADDR(localhost) CONTROL(QMGR) PORT(1417) REPLACE
START LISTENER(1417)
ALTER QMGR REPOS(SALES)
DEFINE CHANNEL(SALES.SALER1) CHLTYPE(CLUSSDR) CONNAME('localhost(1416)')
CLUSTER(SALES) REPLACE
DEFINE CHANNEL(SALES.SALER2) CHLTYPE(CLUSRCVR) CONNAME('localhost(1417)')
CLUSTER(SALES) REPLACE
```

salesrv.txt

```
DEFINE LISTENER(1419) TRPTYPE(TCP) IPADDR(localhost) CONTROL(QMGR) PORT(1419) REPLACE
START LISTENER(1419)
```

```
DEFINE CHANNEL(SALES.SALER1) CHLTYPE(CLUSSDR) CONNAME('localhost(1416)')
CLUSTER(SALES) REPLACE
DEFINE CHANNEL(SALES.SALESRV) CHLTYPE(CLUSRCVR) CONNAME('localhost(1419)')
CLUSTER(SALES) REPLACE
DEFINE QLOCAL(SALES) CLUSTER(SALES) TRIGGER INITQ(SYSTEM.DEFAULT.INITIATION.QUEUE)
PROCESS(ECHO) REPLACE
DEFINE PROCESS(ECHO) APPLICID(AMQSECH) REPLACE
```

gate.txt

```
DEFINE LISTENER(1420) TRPTYPE(TCP) IPADDR(LOCALHOST) CONTROL(QMGR) PORT(1420) REPLACE
START LISTENER(1420)
DEFINE NAMELIST(ALL) NAMES(SALES, FINANCE)
DEFINE CHANNEL(FINANCE.FINR1) CHLTYPE(CLUSSDR) CONNAME('LOCALHOST(1414)')
CLUSTER(FINANCE) REPLACE
DEFINE CHANNEL(FINANCE.GATE) CHLTYPE(CLUSRCVR) CONNAME('LOCALHOST(1420)')
CLUSTER(FINANCE) REPLACE
DEFINE CHANNEL(SALES.SALER1) CHLTYPE(CLUSSDR) CONNAME('LOCALHOST(1416)')
CLUSTER(SALES) REPLACE
DEFINE CHANNEL(SALES.GATE)  CHLTYPE(CLUSRCVR) CONNAME('LOCALHOST(1420)')
CLUSTER(SALES) REPLACE
DEFINE QALIAS(A.SALES) CLUSNL(ALL) TARGET(SALES) TARGTYPE(QUEUE) DEFBIND(NOTFIXED)
REPLACE
DEFINE QREMOTE(FINCLT) RNAME(' ') RQMNAME(FINCLT) CLUSNL(ALL) REPLACE
DEFINE QREMOTE(SALESRV) RNAME(' ') RQMNAME(SALESRV) CLUSNL(ALL) REPLACE
```

2. Test the configuration by running the sample request program.
   a) Start the trigger monitor program on the SALESRV queue manager

      On Windows, open a command window and run the command `runmqtrm -m SALESRV`

   b) Run the sample request program, and send a request.

      On Windows, open a command window and run the command `amqsreq A.SALES FINCLT`

      The request message is echoed back, and after 15 seconds the sample program finishes.
3. Create the definitions to isolate the SALES queue in the `Q.SALES` cluster and separate cluster messages for the SALES and FINANCE cluster on the gateway queue manager.

   Run the command: `runmqsc` *QmgrName* < *filename* where the files are listed in the following list, and the file name almost matches the queue manager name.

   Changes to isolate the sales queue in a new cluster and separate the gateway cluster transmission queues
   chgsaler1.txt

```
DEFINE NAMELIST(CLUSTERS) NAMES(SALES, Q.SALES)
ALTER QMGR REPOS(' ') REPOSNL(CLUSTERS)
DEFINE CHANNEL(Q.SALES.SALER2) CHLTYPE(CLUSSDR) CONNAME('localhost(1417)')
CLUSTER(Q.SALES) REPLACE
DEFINE CHANNEL(Q.SALES.SALER1) CHLTYPE(CLUSRCVR) CONNAME('localhost(1416)')
CLUSTER(Q.SALES) REPLACE
```

   chgsaler2.txt

```
DEFINE NAMELIST(CLUSTERS) NAMES(SALES, Q.SALES)
ALTER QMGR REPOS(' ') REPOSNL(CLUSTERS)
DEFINE CHANNEL(Q.SALES.SALER1) CHLTYPE(CLUSSDR) CONNAME('localhost(1416)')
CLUSTER(Q.SALES) REPLACE
DEFINE CHANNEL(Q.SALES.SALER2) CHLTYPE(CLUSRCVR) CONNAME('localhost(1417)')
CLUSTER(Q.SALES) REPLACE
```

   chgsalesrv.txt

```
DEFINE NAMELIST (CLUSTERS) NAMES(SALES, Q.SALES)
DEFINE CHANNEL(Q.SALES.SALER1) CHLTYPE(CLUSSDR) CONNAME('localhost(1416)')
CLUSTER(Q.SALES) REPLACE
DEFINE CHANNEL(Q.SALES.SAVESRV) CHLTYPE(CLUSRCVR) CONNAME('localhost(1419)')
CLUSTER(Q.SALES) REPLACE
ALTER QLOCAL (SALES) CLUSTER(' ') CLUSNL(CLUSTERS)
```

chggate.txt

```
ALTER NAMELIST(ALL) NAMES(SALES, FINANCE, Q.SALES)
ALTER QMGR DEFCLXQ(CHANNEL)
DEFINE CHANNEL(Q.SALES.SALER1) CHLTYPE(CLUSSDR)  CONNAME('localhost(1416)')
CLUSTER(Q.SALES) REPLACE
DEFINE CHANNEL(Q.SALES.GATE)  CHLTYPE(CLUSRCVR) CONNAME('localhost(1420)')
CLUSTER(Q.SALES) REPLACE
DEFINE QLOCAL (XMITQ.Q.SALES.SALESRV) USAGE(XMITQ) CLCHNAME(Q.SALES.SALESRV) REPLACE
DEFINE QLOCAL (XMITQ.SALES)     USAGE(XMITQ) CLCHNAME(SALES.*)  REPLACE
DEFINE QLOCAL (XMITQ.FINANCE)    USAGE(XMITQ) CLCHNAME(FINANCE.*) REPLACE
```

4. Remove the SALES queue from the SALES cluster.

   Run the **MQSC** command in Figure 15 on page 59:

   ```
   ALTER QLOCAL(SALES) CLUSTER('Q.SALES') CLUSNL(' ')
   ```

   Figure 15. Remove the sales queue on queue manager SALESRV from the sales cluster

5. Switch the channels to the new transmission queues.

   The requirement is to stop and start all the channels that the GATE queue manager is using. To do this with the least number of commands, stop and start the queue manager

   ```
   endmqm -i GATE
   strmqm GATE
   ```

What to do next

1. Rerun the sample request program to verify the new configuration works; see step “2” on page 58
2. Monitor the messages flowing through all the cluster transmission queues on the GATE queue manager:
   a. Alter the definition of each of the cluster transmission queues to turn queue monitoring on.

      ```
      ALTER QLOCAL(SYSTEM.CLUSTER.TRANSMIT.
      name) STATQ(ON)
      ```

   b. Check queue manager statistics monitoring is OFF, to minimize output, and set the monitoring interval to a lower value to perform multiple tests conveniently.

      ```
      ALTER QMGR STATINT(60) STATCHL(OFF) STATQ(OFF) STATMQI(OFF) STATACLS(OFF)
      ```

   c. Restart the GATE queue manager.
   d. Run the sample request program a few times to verify that an equal number of messages are flowing through SYSTEM.CLUSTER.TRANSMIT.Q.SALES.SALESRV and SYSTEM.CLUSTER.TRANSMIT.QUEUE. Requests flow through SYSTEM.CLUSTER.TRANSMIT.Q.SALES.SALESRV and replies through SYSTEM.CLUSTER.TRANSMIT.QUEUE.

      ```
      amqsmon -m GATE -t statistics
      ```

   e. The results over a couple of intervals are as follows:

```
C:\Documents and Settings\Admin>amqsmon -m GATE -t statistics
MonitoringType: QueueStatistics
QueueManager: 'GATE'
IntervalStartDate: '2012-02-27'
IntervalStartTime: '14.59.20'
IntervalEndDate: '2012-02-27'
IntervalEndTime: '15.00.20'
CommandLevel: 700
```

```
ObjectCount: 2
QueueStatistics: 0
QueueName: 'SYSTEM.CLUSTER.TRANSMIT.QUEUE'
CreateDate: '2012-02-24'
CreateTime: '15.58.15'
...
Put1Count: [0, 0]
Put1FailCount: 0
PutBytes: [435, 0]
GetCount: [1, 0]
GetBytes: [435, 0]
...
QueueStatistics: 1
QueueName: 'SYSTEM.CLUSTER.TRANSMIT.Q.SALES.SAVESRV'
CreateDate: '2012-02-24'
CreateTime: '16.37.43'
...
PutCount: [1, 0]
PutFailCount: 0
Put1Count: [0, 0]
Put1FailCount: 0
PutBytes: [435, 0]
GetCount: [1, 0]
GetBytes: [435, 0]
...
MonitoringType: QueueStatistics
QueueManager: 'GATE'
IntervalStartDate: '2012-02-27'
IntervalStartTime: '15.00.20'
IntervalEndDate: '2012-02-27'
IntervalEndTime: '15.01.20'
CommandLevel: 700
ObjectCount: 2
QueueStatistics: 0
QueueName: 'SYSTEM.CLUSTER.TRANSMIT.QUEUE'
CreateDate: '2012-02-24'
CreateTime: '15.58.15'
...
PutCount: [2, 0]
PutFailCount: 0
Put1Count: [0, 0]
Put1FailCount: 0
PutBytes: [863, 0]
GetCount: [2, 0]
GetBytes: [863, 0]
...
QueueStatistics: 1
QueueName: 'SYSTEM.CLUSTER.TRANSMIT.Q.SALES.SAVESRV'
CreateDate: '2012-02-24'
CreateTime: '16.37.43'
...
PutCount: [2, 0]
PutFailCount: 0
Put1Count: [0, 0]
Put1FailCount: 0
PutBytes: [863, 0]
GetCount: [2, 0]
GetBytes: [863, 0]
```

```
...
2 Records Processed.
```

One request and reply message were sent in the first interval and two in the second. You can infer that the request messages were placed on SYSTEM.CLUSTER.TRANSMIT.Q.SALES.SAVESRV, and the reply messages on SYSTEM.CLUSTER.TRANSMIT.QUEUE.

Clustering: Switching cluster transmission queues
Plan how the changes to the cluster transmission queues of an existing production queue manager are going to be brought into effect.

Before you begin

If you reduce the number of messages the switching process has to transfer to the new transmission queue, switching completes more quickly. Read How the process to switch cluster-sender channel to a different transmission queue works for the reasons for trying to empty the transmission queue before proceeding any further.

About this task

You have a choice of two ways of making the changes to cluster transmission queues take effect.

1. Let the queue manager make the changes automatically. This is the default. The queue manager switches cluster-sender channels with pending transmission queue changes when a cluster-sender channel next starts.
2. Make the changes manually. You can make the changes to a cluster-sender channel when it is stopped. You can switch it from one cluster transmission queue to another before the cluster-sender channel starts.

What factors do you take into account when deciding which of the two options to choose, and how do you manage the switch?

Procedure

- Option 1: Let the queue manager make the changes automatically; see "Switching active cluster-sender channels to another set of cluster-transmission queues" on page 62.

  Choose this option if you want the queue manager to make the switch for you.

  An alternative way to describe this option is to say the queue manager switches a cluster-sender channel without you forcing the channel to stop. You do have the option of forcing the channel to stop, and then starting the channel, to make the switch happen sooner. The switch starts when the channel starts, and runs while the channel is running, which is different to option 2. In option 2, the switch takes place when the channel is stopped.

  If you choose this option by letting the switch happen automatically, the switching process starts when a cluster-sender channel starts. If the channel is not stopped, it starts after it becomes inactive, if there is a message to process. If the channel is stopped, start it with the START CHANNEL command.

  The switch process completes as soon as there are no messages left for the cluster-sender channel on the transmission queue the channel was serving. As soon as that is the case, newly arrived messages for the cluster-sender channel are stored directly on the new transmission queue. Until then, messages are stored on the old transmission queue, and the switching process transfers messages from the old transmission queue to the new transmission queue. The cluster-sender channel forwards messages from the new cluster transmission queue during the whole switching process.

  When the switch process completes depends on the state of the system. If you are making the changes in a maintenance window, assess beforehand whether the switching process will complete in time. Whether it will complete in time depends on whether the number of messages that are awaiting transfer from the old transmission queue reaches zero.

The advantage of the first method is it is automatic. A disadvantage is that if the time to make the configuration changes is limited to a maintenance window, you must be confident that you can control the system to complete the switch process inside the maintenance window. If you cannot be sure, option 2 might be a better choice.

- Option 2: Make the changes manually; see "Switching a stopped cluster-sender channel to another cluster transmission queue" on page 63.

  Choose this option if you want to control the entire switching process manually, or if you want to switch a stopped or inactive channel. It is a good choice, if you are switching a few cluster-sender channels, and you want to do the switch during a maintenance window.

  An alternative description of this option is to say that you switch the cluster-sender channel, while the cluster-sender channel is stopped.

  If you choose this option you have complete control over when the switch takes place.
  You can be certain about completing the switching process in a fixed amount of time, within a maintenance window. The time the switch takes depends on how many messages have to be transferred from one transmission queue to the other. If messages keep arriving, it might take a time for the process to transfer all the messages.
  You have the option of switching the channel without transferring messages from the old transmission queue. The switch is "instant".
  When you restart the cluster-sender channel, it starts processing messages on the transmission queue you newly assigned to it.

  The advantage of the second method is you have control over the switching process. The disadvantage is that you must identify the cluster-sender channels to be switched, run the necessary commands, and resolve any in-doubt channels that might be preventing the cluster-sender channel stopping.

Related concepts

How to choose what type of cluster transmission queue to use
How to choose between different cluster transmission queue configuration options.

How the process to switch cluster-sender channel to a different transmission queue works

Related tasks

Clustering: Example configuration of multiple cluster transmission queues
In this task you apply the steps to plan multiple cluster transmission queues to three overlapping clusters. The requirements are to separate messages flows to one cluster queue, from all other message flows, and to store messages for different clusters on different cluster transmission queues.

Switching active cluster-sender channels to another set of cluster-transmission queues
This task gives you three options for switching active cluster-sender channels. One option is to let the queue manager make the switch automatically, which does not affect running applications. The other options are to stop and start channels manually, or to restart the queue manager.

Before you begin

Change the cluster transmission queue configuration. You can change the **DEFCLXQ** queue manager attribute, or add or modify the **CLCHNAME** attribute of transmission queues.

If you reduce the number of messages the switching process has to transfer to the new transmission queue, switching completes more quickly. Read How the process to switch cluster-sender channel to a different transmission queue works for the reasons for trying to empty the transmission queue before proceeding any further.

About this task

Use the steps in the task as a basis for working out your own plan for making cluster-transmission queue configuration changes.

Procedure

1. Optional: Record the current channel status

   Make a record of the status of current and saved channels that are serving cluster transmission queues. The following commands display the status associated with system cluster transmission queues. Add your own commands to display the status associated with cluster-transmission queues that you have defined. Use a convention, such as `XMITQ.` *`ChannelName`*, to name cluster transmission queues that you define to make it easy to display the channel status for those transmission queues.

   ```
   DISPLAY CHSTATUS(*) WHERE(XMITQ LK 'SYSTEM.CLUSTER.TRANSMIT.*')
   DISPLAY CHSTATUS(*) SAVED WHERE(XMITQ LK 'SYSTEM.CLUSTER.TRANSMIT.*')
   ```

2. Switch transmission queues.

   - Do nothing. The queue manager switches cluster-sender channels when they restart after being stopped or inactive.

     Choose this option if you have no rules or concerns about altering a queue manager configuration. Running applications are not affected by the changes.
   - Restart the queue manager. All cluster-sender channels are stopped and restarted automatically on demand.

     Choose this option to initiate all the changes immediately. Running applications are interrupted by the queue manager as it shuts down and restarts.
   - Stop individual cluster-sender channels and restart them.

     Choose this option to change a few channels immediately. Running applications experience a short delay in message transfer between your stopping and starting the message channel again. The cluster-sender channel remains running, except during the time you stopped it. During the switch process messages are delivered to the old transmission queue, transferred to the new transmission queue by the switching process, and forwarded from the new transmission queue by the cluster-sender channel.

3. Optional: Monitor the channels as they switch

   Display the channel status and the transmission queue depth during the switch. The following example display the status for system cluster transmission queues.

   ```
   DISPLAY CHSTATUS(*) WHERE(XMITQ LK 'SYSTEM.CLUSTER.TRANSMIT.*')
   DISPLAY CHSTATUS(*) SAVED WHERE(XMITQ LK 'SYSTEM.CLUSTER.TRANSMIT.*')
   DISPLAY QUEUE('SYSTEM.CLUSTER.TRANSMIT.*') CURDEPTH
   ```

4. Optional: Monitor the messages AMQ7341 The `transmission queue for channel` *`ChannelName`* `switched from queue` *`QueueName`* `to` *`QueueName`* that are written to the queue manager error log.

Switching a stopped cluster-sender channel to another cluster transmission queue
If you choose to make changes manually, you make the changes to a cluster-sender channel when it is stopped and switch it from one cluster transmission queue to another before the cluster-sender channel starts.

Before you begin

You might make some configuration changes, and now want to make them effective without starting the cluster-sender channels that are affected. Alternatively, you make the configuration changes you require as one of the steps in the task.

If you reduce the number of messages the switching process has to transfer to the new transmission queue, switching completes more quickly. Read How the process to switch cluster-sender channel to a different transmission queue works for the reasons for trying to empty the transmission queue before proceeding any further.

About this task

This task switches the transmission queues served by stopped or inactive cluster-sender channels. You might do this task because a cluster-sender channel is stopped, and you want to switch its transmission queue immediately. For example, for some reason a cluster-sender channel is not starting, or has some other configuration problem. To resolve the problem, you decide to create a cluster-sender channel, and associate the transmission queue for the old cluster-sender channel with the new cluster-sender channel you defined.

A more likely scenario is you want to control when reconfiguration of cluster transmission queues is performed. To fully control the reconfiguration, you stop the channels, change the configuration, and then switch the transmission queues.

Procedure

1. Stop the channels that you intend to switch
   a) Stop any running or inactive channels that you intend to switch. Stopping an inactive cluster-sender channel prevents it starting while you are making configuration changes.

   ```
   STOP CHANNEL(ChannelName) MODE(QUIESCE) STATUS(STOPPED)
   ```

2. Optional: Make the configuration changes.

   For example, see “Clustering: Example configuration of multiple cluster transmission queues” on page 53.
3. Switch the cluster-sender channels to the new cluster transmission queues.

   Multi On Multiplatforms, issue the following command:

   ```
   runswchl -m QmgrName -c ChannelName
   ```

   z/OS On z/OS, use the SWITCH function of the CSQUTIL command to switch the messages or monitor what is happening. Use the following command.

   ```
   SWITCH CHANNEL(channel_name) MOVEMSGS(YES)
   ```

   For more information, see SWITCH function.

   The **runswchl**, or CSQUTIL SWITCH, command transfers any messages on the old transmission queue to the new transmission queue. When the number of messages on the old transmission queue for this channel reaches zero, the switch is completed. The command is synchronous. The command writes progress messages to the window during the switching process.

   During the transfer phase existing and new messages destined for the cluster-sender channel are transferred in order to the new transmission queue.

   Because the cluster-sender channel is stopped, the messages build up on the new transmission queue. Contrast the stopped cluster-sender channel, to step “2” on page 63 in “Switching active cluster-sender channels to another set of cluster-transmission queues” on page 62. In that step, the cluster-sender channel is running, so messages do not necessarily build up on the new transmission queue.
4. Optional: Monitor the channels as they switch

   In a different command window, display the transmission queue depth during the switch. The following example display the status for system cluster transmission queues.

   ```
   DISPLAY QUEUE('SYSTEM.CLUSTER.TRANSMIT.*') CURDEPTH
   ```

5. Optional: Monitor the messages AMQ7341 The `transmission queue for channel` *ChannelName* `switched from queue` *QueueName* `to` *QueueName* that are written to the queue manager error log.
6. Restart the cluster-sender channels that you stopped.

The channels do not start automatically, as you stopped them, placing them into STOPPED status.

```
START CHANNEL(ChannelName)
```

Related reference
runswchl
RESOLVE CHANNEL
STOP CHANNEL

Clustering: Migration and modification best practices
This topic provides guidance for planning and administering IBM MQ clusters. This information is a guide based on testing and feedback from customers.

1. "Moving objects in a cluster" on page 65 (Best practices for moving objects around inside a cluster, without installing any fix packs or new versions of IBM MQ).
2. "Upgrades and maintenance installations" on page 66 (Best practices for keeping a working cluster architecture up and running, while applying maintenance or upgrades and testing the new architecture).

Moving objects in a cluster

Applications and their queues

When you must move a queue instance hosted on one queue manager to be hosted on another, you can work with the workload balancing parameters to ensure a smooth transition.

Create an instance of the queue where it is to be newly hosted, but use cluster workload balancing settings to continue sending messages to the original instance until your application is ready to switch. This is achieved with the following steps:

1. Set the **CLWLRANK** property of the existing queue to a high value, for example five.
2. Create the new instance of the queue and set its **CLWLRANK** property to zero.
3. Complete any further configuration of the new system, for example deploy and start consuming applications against the new instance of the queue.
4. Set the **CLWLRANK** property of the new queue instance to be higher than the original instance, for example nine.
5. Allow the original queue instance to process any queued messages in the system and then delete the queue.

Moving entire queue managers

If the queue manager is staying on the same host, but the IP address is changing, then the process is as follows:

- DNS, when used correctly, can help simplify the process. For information about using DNS by setting the Connection name (CONNAME) channel attribute, see ALTER CHANNEL.
- If moving a full repository, ensure that you have at least one other full repository which is running smoothly (no problems with channel status for example) before making changes.
- Suspend the queue manager using the SUSPEND QMGR command to avoid traffic buildup.
- Modify the IP address of the computer. If your CLUSRCVR channel definition uses an IP address in the CONNAME field, modify this IP address entry. The DNS cache might need to be flushed through to ensure that updates are available everywhere.
- When the queue manager reconnects to the full repositories, channel auto-definitions automatically resolve themselves.
- If the queue manager hosted a full repository and the IP address changes, it is important to ensure that partials are switched over as soon as possible to point any manually defined CLUSSDR channels to the new location. Until this switch is carried out, these queue managers might be able to only

contact the remaining (unchanged) full repository, and warning messages might be seen regarding the incorrect channel definition.

- Resume the queue manager using the RESUME QMGR command.

If the queue manager must be moved to a new host, it is possible to copy the queue manager data and restore from a backup. This process is not recommended however, unless there are no other options; it might be better to create a queue manager on a new machine and replicate queues and applications as described in the previous section. This situation gives a smooth rollover/rollback mechanism.

If you are determined to move a complete queue manager using backup, follow these best practices:

- Treat the whole process as a queue manager restore from backup, applying any processes you would usually use for system recovery as appropriate for your operating system environment.
- Use the **REFRESH CLUSTER** command after migration to discard all locally held cluster information (including any auto-defined channels that are in doubt), and force it to be rebuilt.

  Note: For large clusters, using the **REFRESH CLUSTER** command can be disruptive to the cluster while it is in progress, and again at 27 day intervals thereafter when the cluster objects automatically send status updates to all interested queue managers. See Refreshing in a large cluster can affect performance and availability of the cluster.

When creating a queue manager and replicating the setup from an existing queue manager in the cluster (as described previously in this topic), never treat the two different queue managers as actually being the same. In particular, do not give a new queue manager the same queue manager name and IP address. Attempting to 'drop in' a replacement queue manager is a frequent cause of problems in IBM MQ clusters. The cache expects to receive updates including the **QMID** attribute, and state can be corrupted.

If two different queue managers are accidentally created with the same name, it is recommended to use the RESET CLUSTER **QMID** command to eject the incorrect entry from the cluster.

Upgrades and maintenance installations

Avoid the so-called big bang scenario (for example, stopping all cluster and queue manager activity, applying all upgrades and maintenance to all queue managers, then starting everything at the same time). Clusters are designed to still work with multiple versions of queue manager coexisting, so a well-planned, phased maintenance approach is recommended.

Have a backup plan:

- Have you taken backups?
- Avoid using new cluster functionality immediately: Wait until you are sure that all the queue managers are upgraded to the new level, and are certain that you are not going to roll any of them back. Using new cluster function in a cluster where some queue managers are still at an earlier level can lead to undefined behavior.

  A repository stores a record it receives in its own version. If the record it receives is at a later version, the later version attributes are discarded when the record is stored. An IBM MQ 9.4 queue manager receiving information about an IBM MQ 10.0 queue manager stores only IBM MQ 9.4 information. An IBM MQ 10.0 repository receiving an IBM MQ 9.4 record stores default values for attributes introduced in the later version. The defaults define the values for the attributes that are not included in the record it receives.

Migrate full repositories first. Although they can forward information that they do not understand, they cannot persist it, so it is not the recommended approach unless absolutely necessary. For more information, see Queue manager cluster migration.

Clustering: Using REFRESH CLUSTER best practices
You use the **REFRESH CLUSTER** command to discard all locally held information about a cluster and rebuild that information from the full repositories in the cluster. You should not need to use this

command, except in exceptional circumstances. If you do need to use it, there are special considerations about how you use it. This information is a guide based on testing and feedback from customers.

Only run REFRESH CLUSTER if you really need to do so

The IBM MQ cluster technology ensures that any change to the cluster configuration, such as a change to a clustered queue, automatically becomes known to any member of the cluster that needs to know the information. There is no need for further administrative steps to be taken to achieve this propagation of information.

If such information does not reach the queue managers in the cluster where it is required, for example a clustered queue is not known by another queue manager in the cluster when an application attempts to open it for the first time, it implies a problem in the cluster infrastructure. For example, it is possible that a channel cannot be started between a queue manager and a full repository queue manager. Therefore, any situation where inconsistencies are observed must be investigated. If possible, resolve the situation without using the **REFRESH CLUSTER** command.

In rare circumstances that are documented elsewhere in this product documentation, or when requested by IBM Support, you can use the **REFRESH CLUSTER** command to discard all locally held information about a cluster and rebuild that information from the full repositories in the cluster.

Refreshing in a large cluster can affect performance and availability of the cluster

Use of the **REFRESH CLUSTER** command can be disruptive to the cluster while it is in progress, for example by creating a sudden increase in work for the full repositories as they process the repropagation of queue manager cluster resources. If you are refreshing in a large cluster (that is, many hundreds of queue managers) you should avoid use of the command in day-to-day work if possible and use alternative methods to correct specific inconsistencies. For example, if a cluster queue is not being correctly propagated across the cluster, an initial investigation technique of updating the clustered queue definition, such as altering its description, repropagates the queue configuration across the cluster. This process can help to identify the problem and potentially resolve a temporary inconsistency.

If alternative methods cannot be used, and you have to run **REFRESH CLUSTER** in a large cluster, you should do so at off-peak times or during a maintenance window to avoid impact on user workloads. You should also avoid refreshing a large cluster in a single batch, and instead stagger the activity as explained in “Avoid performance and availability issues when cluster objects send automatic updates” on page 67.

Avoid performance and availability issues when cluster objects send automatic updates

After a new cluster object is defined on a queue manager, an update for this object is generated every 27 days from the time of definition, and sent to every full repository in the cluster and onwards to any other interested queue managers. When you issue the **REFRESH CLUSTER** command to a queue manager, you reset the clock for this automatic update on all objects defined locally in the specified cluster.

If you refresh a large cluster (that is, many hundreds of queue managers) in a single batch, or in other circumstances such as recreating a system from configuration backup, after 27 days all of those queue managers will re-advertise all of their object definitions to the full repositories at the same time. This could again cause the system to run significantly slower, or even become unavailable, until all the updates have completed. Therefore, when you have to refresh or re-create multiple queue managers in a large cluster, you should stagger the activity over several hours, or several days, so that subsequent automatic updates do not regularly impact system performance.

The system cluster history queue

When a **REFRESH CLUSTER** is performed, the queue manager takes a snapshot of the cluster state before the refresh and stores it on the SYSTEM.CLUSTER.HISTORY.QUEUE (SCHQ) if it is defined on the queue manager. This snapshot is for IBM service purposes only, in case of later problems with the system.

The SCHQ is defined by default on distributed queue managers on startup. For z/OS migration, the SCHQ must be manually defined.

Messages on the SCHQ expire after three months.

Related concepts

"REFRESH CLUSTER considerations for publish/subscribe clusters" on page 101
Issuing the **REFRESH CLUSTER** command results in the queue manager temporarily discarding locally held information about a cluster, including any cluster topics and their associated proxy subscriptions.

Related reference

Application issues seen when running REFRESH CLUSTER

MQSC Commands reference: REFRESH CLUSTER

Clustering: Availability, multi-instance, and disaster recovery
This topic provides guidance for planning and administering IBM MQ clusters. This information is a guide based on testing and feedback from customers.

IBM MQ Clustering itself is not a High Availability solution, but in some circumstances it can be used to improve the availability of services using IBM MQ. For guidance on how to combine highly available queue managers with IBM MQ clusters, see Combining highly available queue managers with IBM MQ clusters.

Planning your distributed publish/subscribe network

You can create a network of queue managers where subscriptions created on one queue manager will receive matching messages published by an application connected to another queue manager in the network. To choose a suitable topology, you need to consider your requirements for manual control, network size, frequency of change, availability and scalability.

Before you begin

This task assumes that you understand what distributed publish/subscribe networks are, and how they work. For a technical overview, see Distributed publish/subscribe networks.

About this task

There are three basic topologies for a publish/subscribe network:

- Direct routed cluster
- Topic host routed cluster
- Hierarchy

For the first two topologies, the starting point is an IBM MQ cluster configuration. The third topology can be created with or without a cluster. See"Planning your distributed queues and clusters" on page 19for information about planning the underlying queue manager network.

A *Direct routed cluster* is the simplest topology to configure when a cluster is already present. Any topic that you define on any queue manager is automatically made available on every queue manager in the cluster, and publications are routed directly from any queue manager where a publishing application connects, to each of the queue managers where matching subscriptions exist. This simplicity of configuration relies on IBM MQ maintaining a high level of sharing of information and connectivity between every queue manager in the cluster. For small and simple networks (that is, a small number of queue managers, and a fairly static set of publishers and subscribers) this is acceptable. However, when used in larger or more dynamic environments the overhead might be prohibitive. See"Direct routing in publish/subscribe clusters" on page 73.

A *Topic host routed cluster* gives the same benefit as a direct routed cluster, by making any topic that you define on any queue manager in the cluster automatically available on every queue manager in the cluster. However, topic host routed clusters require you to carefully choose the queue managers that host each topic, because all information and publications for that topic pass through those topic host queue managers. This means that the system does not have to maintain channels and information flows

between all queue managers. However it also means that publications might no longer be sent directly to subscribers, but might be routed through a topic host queue manager. For these reasons additional load might be put on the system, especially on the queue managers hosting the topics, so careful planning of the topology is required. This topology is particularly effective for networks that contain many queue managers, or that host a dynamic set of publishers and subscribers (that is, publishers or subscribers that are frequently added or removed). Additional topic hosts can be defined to improve availability of routes and to horizontally scale publication workload. See“Topic host routing in publish/subscribe clusters” on page 78.

A *Hierarchy* requires the most manual configuration to set up, and is the hardest topology to modify. You must manually configure the relationships between each queue manager in the hierarchy and its direct relations. After relationships are configured, publications will (as for the previous two topologies) be routed to subscriptions on other queue managers in the hierarchy. Publications are routed using the hierarchy relationships. This allows very specific topologies to be configured to suit different requirements, but it can also result in publications requiring many "hops" through intermediate queue managers to reach the subscriptions. There is always only one route through a hierarchy for a publication, so availability of every queue manager is critical. Hierarchies are typically only preferable where a single cluster cannot be configured; for example when spanning multiple organizations. See“Routing in publish/subscribe hierarchies” on page 102.

Where necessary, the above three topologies can be combined to solve specific topographical requirements. For an example, see Combining the topic spaces of multiple clusters.

To choose a suitable topology for your distributed publish/subscribe network, you need to consider the following broad questions:

- How big will your network be?
- How much manual control do you need over its configuration?
- How dynamic will the system be, both in terms of topics and subscriptions, and in terms of queue managers?
- What are your availability and scalability requirements?
- Can all queue managers connect directly to each other?

Procedure

- Estimate how big your network needs to be.
  a) Estimate how many topics you need.
  b) Estimate how many publishers and subscribers you expect to have.
  c) Estimate how many queue managers will be involved in publish/subscribe activities.

  See also“Publish/subscribe clustering: Best practices” on page 87, especially the following sections:

  – How to size your system
  – Reasons to limit the number of cluster queue managers involved in publish/subscribe activity
  – How to decide which topics to cluster

  If your network will have many queue managers, and handle many publishers and subscribers, you probably need to use a topic host routed cluster or a hierarchy. Direct routed clusters require almost no manual configuration, and can be a good solution for small or static networks.
- Consider how much manual control you need over which queue manager hosts each topic, publisher or subscriber.
  a) Consider whether some of your queue managers are less capable than others.
  b) Consider whether the communication links to some of your queue managers are more fragile than to others.
  c) Identify cases where you expect a topic to have many publications and few subscribers.

d) Identify cases where you expect a topic to have many subscribers and few publications.

In all topologies, publications are delivered to subscriptions on other queue managers. In a direct routed cluster those publications take the shortest path to the subscriptions. In a topic host routed cluster or a hierarchy, you control the route that publications take. If your queue managers differ in their capability, or have differing levels of availability and connectivity, you will probably want to assign specific workloads to specific queue managers. You can do this using either a topic host routed cluster or a hierarchy.

In all topologies, co-locating the publishing applications on the same queue manager as the subscriptions whenever possible minimizes overheads and maximizes performance. For topic host routed clusters, consider putting publishers or subscribers on the queue managers that host the topic. This removes any extra "hops" between queue managers to pass a publication to a subscriber. This approach is particularly effective in cases where a topic has many publishers and few subscribers, or many subscribers and few publishers. See, for example, Topic host routing using centralized publishers or subscribers.

See also“Publish/subscribe clustering: Best practices” on page 87, especially the following sections:

– How to decide which topics to cluster
– Publisher and subscription location

- Consider how dynamic the network activity will be.
  a) Estimate how frequently subscribers will be added and removed on different topics.

  Whenever a subscription is added or removed from a queue manager, and it is the first or last subscription for that specific topic string, that information is communicated to other queue managers in the topology. In a direct routed cluster and a hierarchy, this subscription information is propagated to every queue manager in the topology whether or not they have publishers on the topic. If the topology consists of many queue managers, this might be a significant performance overhead. In a topic host routed cluster, this information is only propagated to those queue managers that host a clustered topic that maps to the subscription's topic string.

  See also the Subscription change and dynamic topic strings section of“Publish/subscribe clustering: Best practices” on page 87.

  Note: In very dynamic systems, where the set of many unique topic strings is rapidly and constantly being changed, it might be best to switch the model to a "publish everywhere" mode. See Subscription performance in publish/subscribe networks.

  b) Consider how dynamic the queue managers are in the topology.

  A hierarchy requires each change in queue manager in the topology to be manually inserted or removed from the hierarchy, with care taken when changing queue managers at higher levels in the hierarchy. Queue managers in a hierarchy typically also use manually configured channel connections. You must maintain these connections, adding and removing channels as queue managers are added and removed from the hierarchy.

  In a publish/subscribe cluster, queue managers are automatically connected to any other queue manager that is required when they first join the cluster, and automatically become aware of topics and subscriptions.
- Consider your route availability and publication traffic scalability requirements.
  a) Decide whether you need to always have an available route from a publishing queue manager to a subscribing queue manager, even when a queue manager is unavailable.
  b) Consider how scalable you need the network to be. Decide whether the level of publication traffic is too high to be routed through a single queue manager or channel, and whether that level of publication traffic must be handled by a single topic branch or can be spread across multiple topic branches.
  c) Consider whether you need to maintain message ordering.

  Because a direct routed cluster sends messages directly from publishing queue managers to subscribing queue managers, you do not need to consider the availability of intermediate queue

managers along the route. Similarly, scaling to the intermediate queue managers is not a consideration. However, as previously mentioned, the overhead of automatically maintaining channels and information flows between all queue managers in the cluster can significantly affect performance, especially in a large or dynamic environment.

A topic host routed cluster can be tuned for individual topics. You can ensure that each branch of the topic tree that has a considerable publication workload is defined on a different queue manager, and that each queue manager is sufficiently performant and available for the expected workload for that branch of the topic tree. You can also improve availability and horizontal scaling further by defining each topic on multiple queue managers. This allows the system to route around unavailable topic host queue managers, and to workload balance publication traffic across them. However, when you define a given topic on multiple queue managers, you also introduce the following constraints:

- You lose message ordering across publications.
- You cannot use retained publications. See"Design considerations for retained publications in publish/subscribe clusters" on page 100.

You cannot configure high availability or scalability of routing in a hierarchy through multiple routes.

See also the Publication traffic section of"Publish/subscribe clustering: Best practices" on page 87.

- Based on these calculations, use the links provided to help you decide whether to use a topic host routed cluster, a direct routed cluster, a hierarchy, or a mixture of these topologies.

What to do next

You are now ready to configure your distributed publish/subscribe network.

Related tasks

Configuring a queue manager cluster

Configuring distributed queuing

Configuring a publish/subscribe cluster

Connecting a queue manager to a publish/subscribe hierarchy

Designing publish/subscribe clusters

There are two basic publish/subscribe cluster topologies: *direct routing* and *topic host routing*. Each has different benefits. When you design your publish/subscribe cluster, choose the topology that best fits your expected network requirements.

For an overview of the two publish/subscribe cluster topologies, see Publish/subscribe clusters. To help you evaluate your network requirements, see "Planning your distributed publish/subscribe network" on page 68 and "Publish/subscribe clustering: Best practices" on page 87.

In general, both cluster topologies provide the following benefits:

- Simple configuration on top of a point-to-point cluster topology.
- Automatic handling of queue managers joining and leaving the cluster.
- Ease of scaling for additional subscriptions and publishers, by adding extra queue managers and distributing the additional subscriptions and publishers across them.

However, the two topologies have different benefits as the requirements become more specific.

Direct routed publish/subscribe clusters

With direct routing, any queue manager in the cluster sends publications from connected applications direct to any other queue manager in the cluster with a matching subscription.

A direct routed publish/subscribe cluster provides the following benefits:

- Messages destined for a subscription on a specific queue manager in the same cluster are transported directly to that queue manager and do not need to pass through an intermediate queue manager. This can improve performance in comparison with a topic host routed topology, or a hierarchical topology.

- Because all queue managers are directly connected to each other, there is no single point of failure in the routing infrastructure of this topology. If one queue manager is not available, subscriptions on other queue managers in the cluster are still able to receive messages from publishers on available queue managers.
- It is very simple to configure, especially on an existing cluster.

Things to consider when using a direct routed publish/subscribe cluster:

- All queue managers in the cluster become aware of all other queue managers in the cluster.
- Queue managers in a cluster that host one or more subscriptions to a clustered topic, automatically create cluster sender channels to all other queue managers in the cluster, even when those queue managers are not publishing messages on any clustered topics.
- The first subscription on a queue manager to a topic string under a clustered topic results in a message being sent to every other queue manager in the cluster. Similarly, the last subscription on a topic string to be deleted also results in a message. The more individual topic strings being used under a clustered topic, and the higher the rate of change of subscriptions, the more inter-queue manager communication occurs.
- Every queue manager in the cluster maintains the knowledge of subscribed topic strings that it is informed of, even when the queue manager is neither publishing nor subscribing to those topics.

For the above reasons, all queue managers in a cluster with a direct routed topic defined will incur an additional overhead. The more queue managers there are in the cluster, the greater the overhead. Likewise the more topic strings subscribed to, and the greater their rate of change, the greater the overhead. This can result in too much load on queue managers running on small systems in a large or dynamic direct routed publish/subscribe cluster. See Direct routed publish/subscribe performance for further information.

When you know that a cluster cannot accommodate the overheads of direct routed clustered publish/subscribe, you can instead use topic host routed publish/subscribe. Alternatively, in extreme situations, you can completely disable clustered publish/subscribe functionality by setting the queue manager attribute **PSCLUS** to DISABLED on every queue manager in the cluster. See “Inhibiting clustered publish/subscribe” on page 97. This prevents any clustered topic from being created, and therefore ensures that your network does not incur any overheads associated with clustered publish/subscribe.

Topic host routed publish/subscribe clusters

With topic host routing, the queue managers where clustered topics are administratively defined become routers for publications. Publications from non-hosting queue managers in the cluster are routed through the hosting queue manager to any queue manager in the cluster with a matching subscription.

A topic host routed publish/subscribe cluster provides the following extra benefits over a direct routed publish/subscribe cluster:

- Only queue managers on which topic host routed topics are defined are made aware of all other queue managers in the cluster.
- Only the topic host queue managers need to be able to connect to all other queue managers in the cluster, and will typically only connect to those where subscriptions exist. Therefore there are significantly fewer channels running between queue managers.
- Cluster queue managers that host one or more subscriptions to a clustered topic automatically create cluster sender channels only to queue managers that host a cluster topic that maps to the topic string of the subscription.
- The first subscription on a queue manager to a topic string under a clustered topic results in a message being sent to a queue manager in the cluster that hosts the clustered topic. Similarly, the last subscription on a topic string to be deleted also results in a message. The more individual topic strings being used under a clustered topic, and the higher the rate of change of subscriptions, the more inter-queue manager communication occurs, but only between subscription hosts and topic hosts.
- More control over the physical configuration. With direct routing, all queue managers have to participate in the publish/subscribe cluster, increasing their overheads. With topic host routing, only the topic host

queue managers are aware of other queue managers and their subscriptions. You explicitly choose the topic host queue managers, therefore you can ensure that those queue managers are running on adequate equipment, and you can use less powerful systems for the other queue managers.

Things to consider when using a topic host routed publish/subscribe cluster:

- An extra "hop" between a publishing queue manager and a subscribing queue manager is introduced when the publisher or the subscriber is not located on a topic hosting queue manager. The latency caused by the extra "hop" can mean that topic host routing is less efficient that direct routing.
- On large clusters, topic host routing eases the significant performance and scaling issues that you can get with direct routing.
- You might choose to define all your topics on a single queue manager, or on a very small number of queue managers. If you do this, make sure the topic host queue managers are hosted on powerful systems with good connectivity.
- You can define the same topic on more than one queue manager. This improves the availability of the topic, and also improves scalability because IBM MQ workload balances publications for a topic across all hosts for that topic. Note, however, that defining the same topic on more than one queue manager loses message order for that topic.
- By hosting different topics on different queue managers, you can improve scalability without losing message order.

Related tasks

Scenario: Creating a publish/subscribe cluster

Configuring a publish/subscribe cluster

Tuning distributed publish/subscribe networks

Troubleshooting distributed publish/subscribe problems

Direct routing in publish/subscribe clusters

Publications from any publishing queue manager are routed direct to any other queue manager in the cluster with a matching subscription.

For an introduction to how messages are routed between queue managers in publish/subscribe hierarchies and clusters, see Distributed publish/subscribe networks.

A direct routed publish/subscribe cluster behaves as follows:

- All queue managers automatically know of all other queue managers.
- All queue managers with subscriptions to clustered topics create channels to all other queue managers in the cluster and inform them of their subscriptions.
- Messages published by an application are routed from the queue manager that it is connected to, direct to each queue manager where a matching subscription exists.

The following diagram shows a queue manager cluster that is not currently used for publish/subscribe or point-to-point activities. Note that every queue manager in the cluster connects only to and from the full repository queue managers.

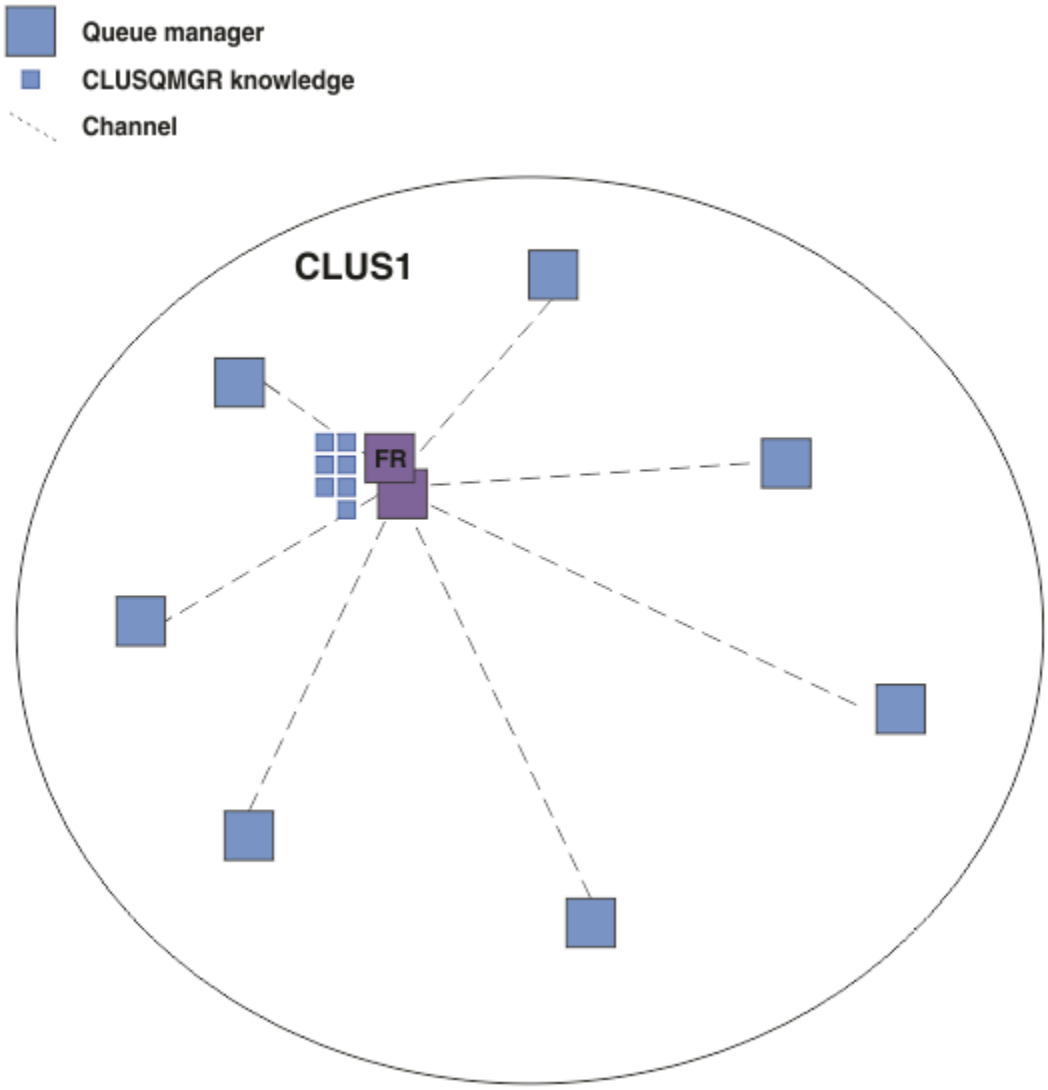


Figure 16. A queue manager cluster

For publications to flow between queue managers in a direct routed cluster, you cluster a branch of the topic tree as described in Configuring a publish/subscribe cluster, and specify *direct routing* (the default).

In a direct routed publish/subscribe cluster, you define the topic object on any queue manager in the cluster. When you do this, knowledge of the object, and knowledge of all other queue managers in the cluster, is automatically pushed to all queue managers in the cluster by the full repository queue managers. This happens before any queue manager references the topic:

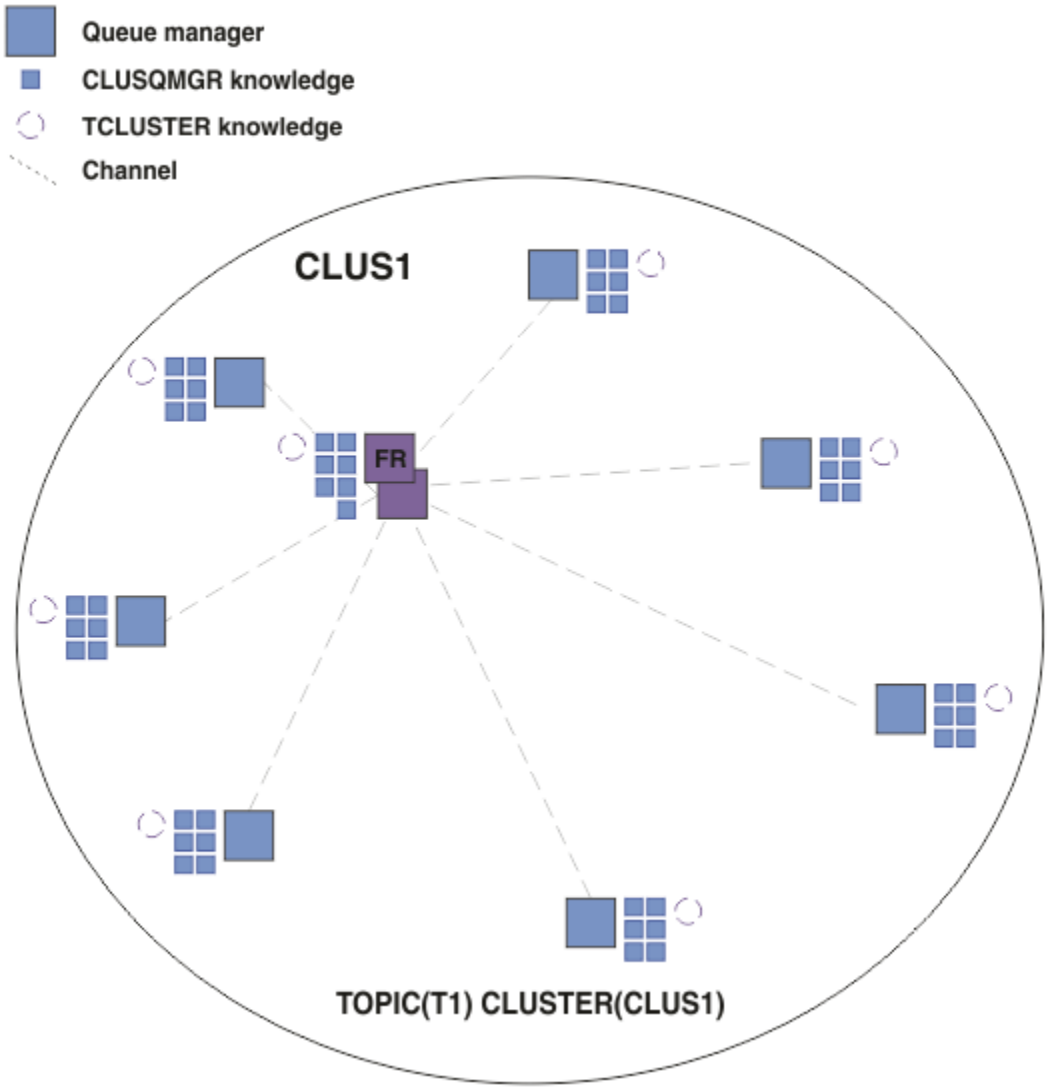


Figure 17. A direct routed publish/subscribe cluster

When a subscription is created, the queue manager that hosts the subscription establishes a channel to every queue manager in the cluster, and sends details of the subscription. This distributed subscription knowledge is represented by a proxy subscription on each queue manager. When a publication is produced on any queue manager in the cluster that matches that proxy subscription's topic string, a cluster channel is established from the publisher queue manager to each queue manager hosting a subscription, and the message is sent to each of them.

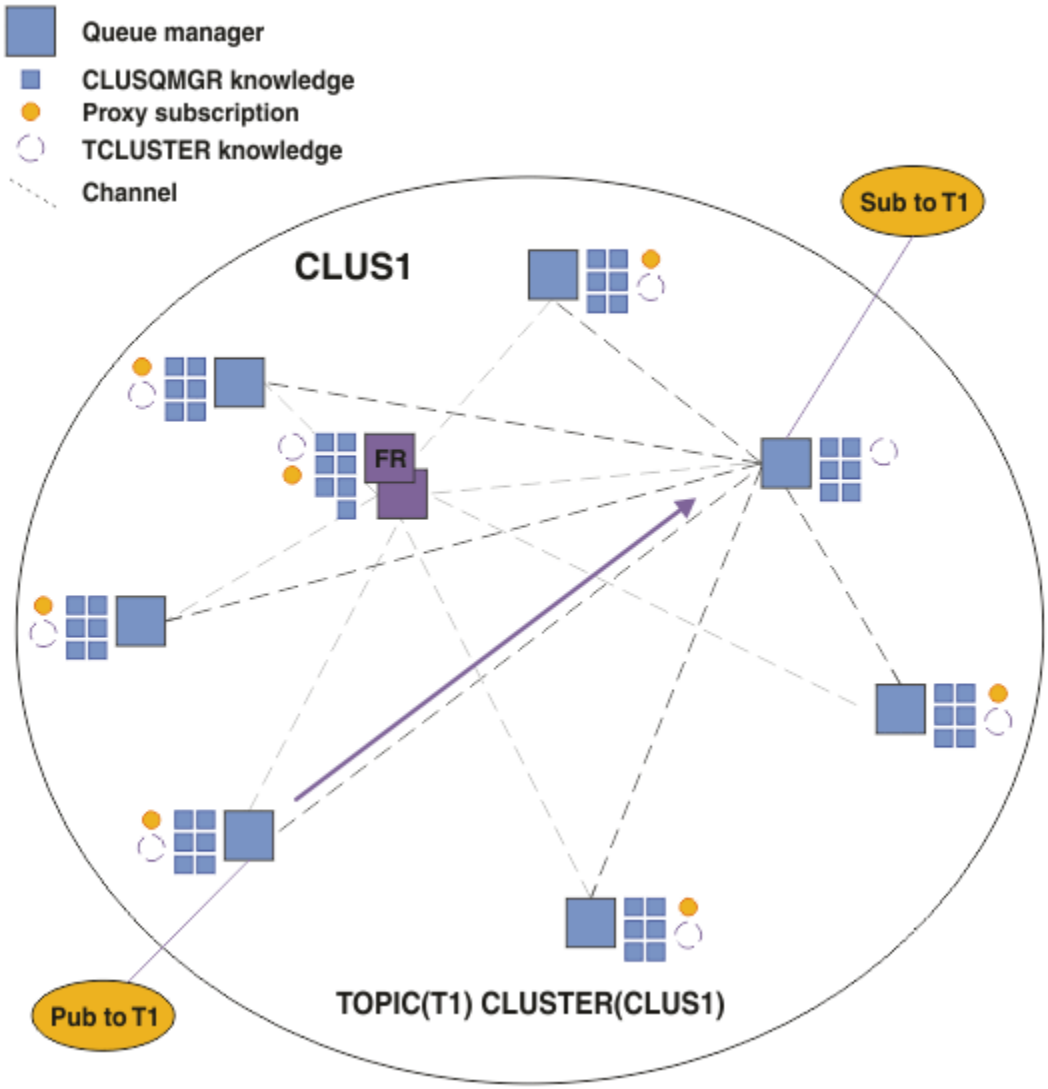


Figure 18. A direct routed publish/subscribe cluster with a publisher and a subscriber to a clustered topic

The direct routing of publications to subscription hosting queue managers simplifies configuration and minimizes the latency in delivering publications to subscriptions.

However, depending on the location of subscriptions and publishers, your cluster can quickly become fully interconnected, with every queue manager having a direct connection to every other queue manager. This might or might not be acceptable in your environment. Similarly, if the set of topic strings being subscribed to is changing frequently, the overhead of propagating that information between all queue managers can also become significant. All queue managers in a direct routed publish/subscribe cluster must be able to cope with these overheads.

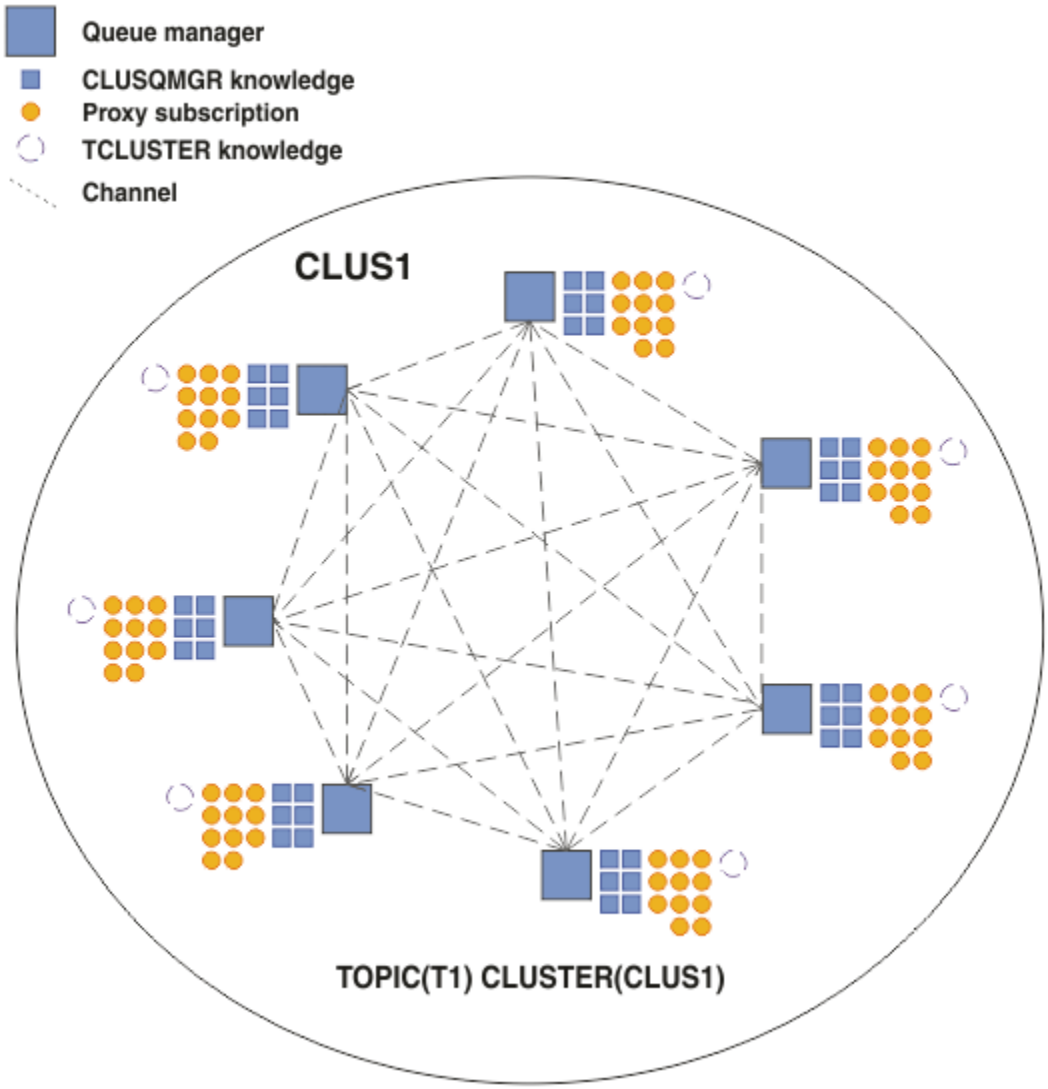


Figure 19. A direct routed publish/subscribe cluster that is fully interconnected

Summary and additional considerations

A direct routed publish/subscribe cluster needs little manual intervention to create or administer, and provides direct routing between publishers and subscribers. For certain configurations it is usually the most appropriate topology, notably clusters with few queue managers, or where high queue manager connectivity is acceptable and subscriptions are changing infrequently. However it also imposes certain constraints upon your system:

- The load on each queue manager is proportional to the total number of queue managers in the cluster. Therefore, in larger clusters, individual queue managers and the system as a whole can experience performance issues.
- By default, all clustered topic strings subscribed to are propagated throughout the cluster, and publications are propagated only to remote queue managers that have a subscription to the associated topic. Therefore rapid changes to the set of subscriptions can become a limiting factor. You can change this default behavior, and instead have all publications propagated to all queue managers, which removes the need for proxy subscriptions. This reduces the subscription knowledge traffic, but is likely to increase the publication traffic and the number of channels each queue manager establishes. See Subscription performance in publish/subscribe networks.

  Note: A similar restriction also applies to hierarchies.

- Because of the interconnected nature of publish/subscribe queue managers, it takes time for proxy subscriptions to propagate around all nodes in the network. Remote publications do not necessarily start being subscribed to immediately, so early publications might not be sent following a subscription to a new topic string. You can remove the problems caused by the subscription delay by having all publications propagated to all queue managers, which removes the need for proxy subscriptions. See Subscription performance in publish/subscribe networks.

  Note: This restriction also applies to hierarchies.

Before you use direct routing, explore the alternative approaches detailed in "Topic host routing in publish/subscribe clusters" on page 78, and "Routing in publish/subscribe hierarchies" on page 102.

Topic host routing in publish/subscribe clusters

Publications from non-hosting queue managers in the cluster are routed through the hosting queue manager to any queue manager in the cluster with a matching subscription.

For an introduction to how messages are routed between queue managers in publish/subscribe hierarchies and clusters, see Distributed publish/subscribe networks.

To understand the behavior and benefits of topic host routing it is best to first understand "Direct routing in publish/subscribe clusters" on page 73.

A topic host routed publish/subscribe cluster behaves as follows:

- Clustered administered topic objects are manually defined on individual queue managers in the cluster. These are referred to as *topic host queue managers*.
- When a subscription is made on a cluster queue manager, channels are created from the subscription host queue manager to the topic host queue managers, and proxy subscriptions are created only on the queue managers that host the topic.
- When an application publishes information to a topic, the connected queue manager always forwards the publication to one queue manager that hosts the topic, which passes it on to all queue managers in the cluster that have matching subscriptions to the topic.

This process is explained in more detail in the following examples.

Topic host routing using a single topic host

For publications to flow between queue managers in a topic host routed cluster, you cluster a branch of the topic tree as described in Configuring a publish/subscribe cluster, and specify *topic host routing*.

There are a number of reasons to define a topic host routed topic object on multiple queue managers in a cluster. However, for simplicity we start with a single topic host.

The following diagram shows a queue manager cluster that is not currently used for publish/subscribe or point-to-point activities. Note that every queue manager in the cluster connects only to and from the full repository queue managers.

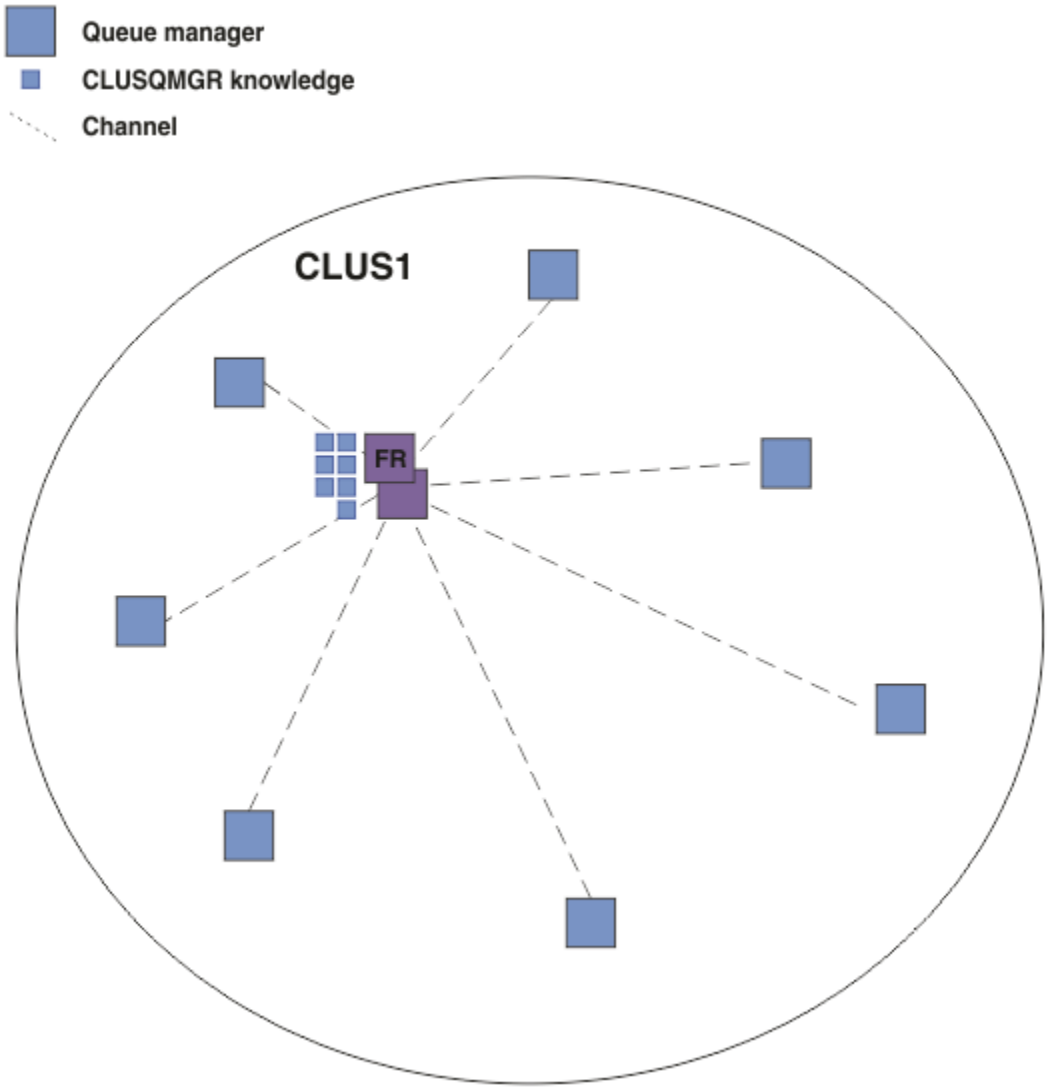


Figure 20. A queue manager cluster

In a topic host routed publish/subscribe cluster, you define the topic object on a specific queue manager in the cluster. Publish/subscribe traffic then flows through that queue manager, making it a critical queue manager in the cluster and increasing its workload. For these reasons it is not recommended to use a full repository queue manager, but to use another queue manager in the cluster. When you define the topic object on the host queue manager, knowledge of the object and its host is automatically pushed, by the full repository queue managers, to all the other queue managers in the cluster. Note that, unlike *direct routing*, each queue manager is not told about every other queue manager in the cluster.

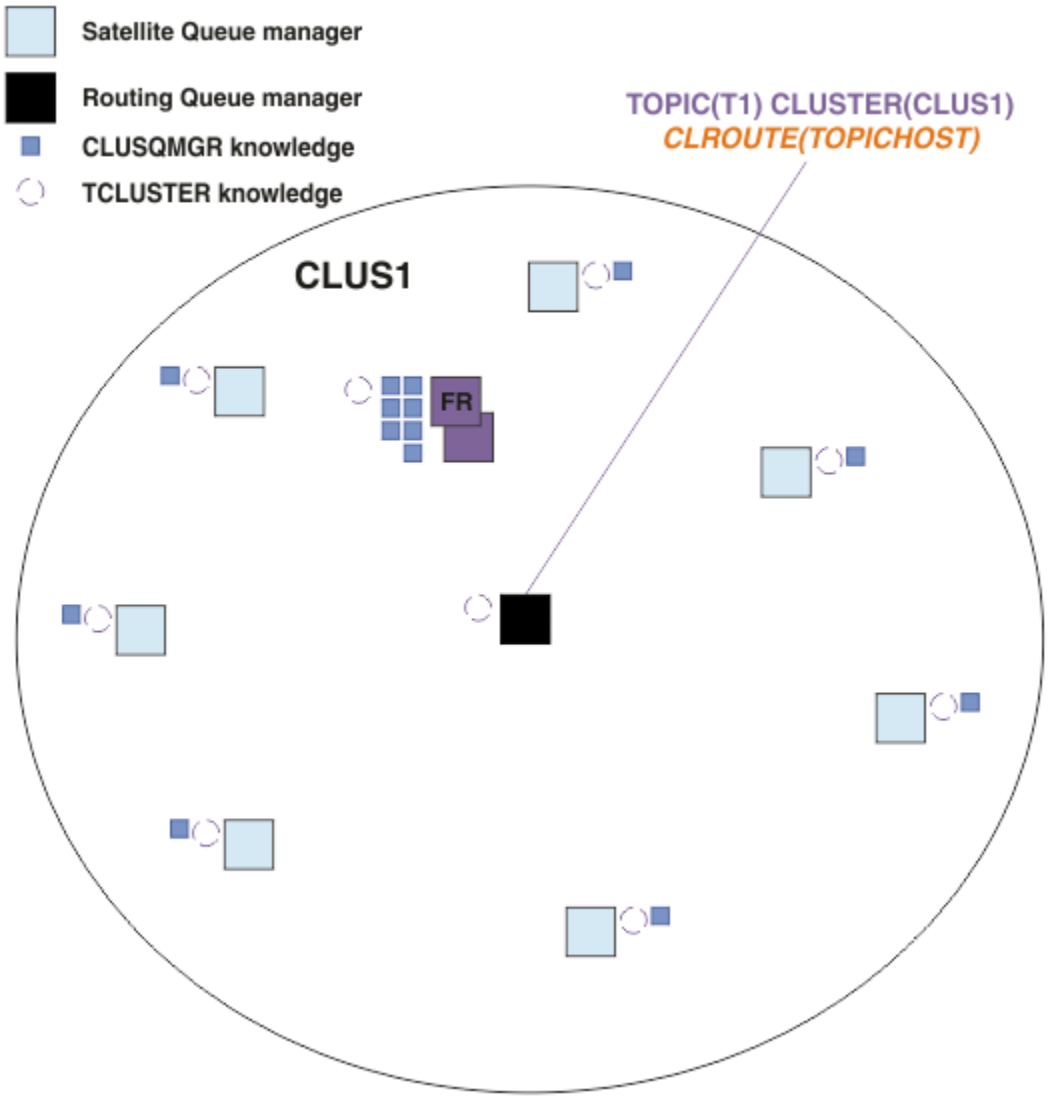


Figure 21. A topic host routed publish/subscribe cluster with one topic defined on one topic host

When a subscription is created on a queue manager, a channel is created between the subscribing queue manager and the topic host queue manager. The subscribing queue manager connects only to the topic host queue manager, and sends details of the subscription (in the form of a *proxy subscription*). The topic host queue manager does not forward this subscription information on to any further queue managers in the cluster.

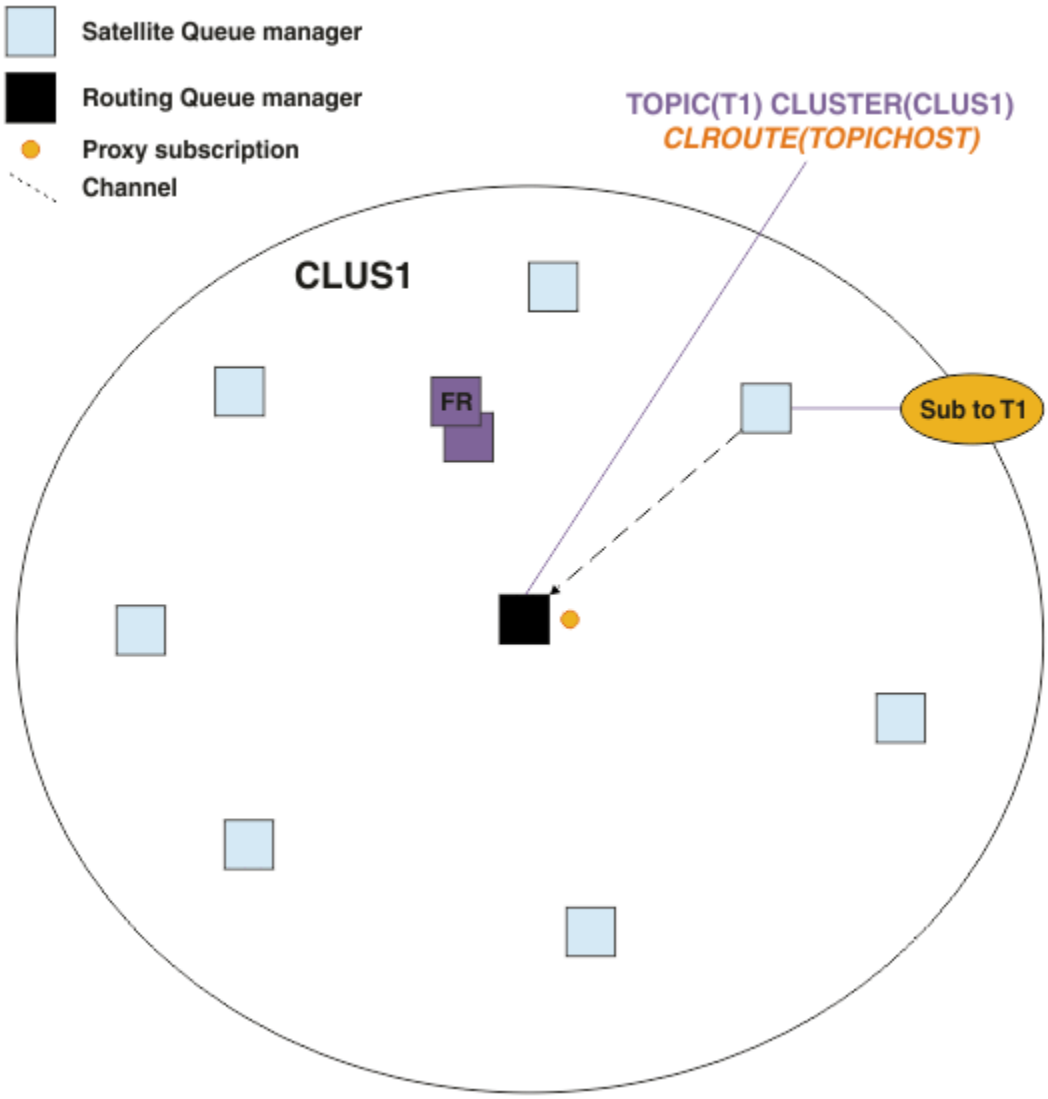


Figure 22. A topic host routed publish/subscribe cluster with one topic defined on one topic host, and one subscriber

When a publishing application connects to another queue manager and a message is published, a channel is created between the publishing queue manager and the topic host queue manager, and the message is forwarded to that queue manager. The publishing queue manager has no knowledge of any subscriptions on other queue managers in the cluster, so the message is forwarded to the topic host queue manager even if there are no subscribers to that topic in the cluster. The publishing queue manager connects only to the topic host queue manager. Publications are routed through the topic host to the subscribing queue managers, if any exist.

Subscriptions on the same queue manager as the publisher are satisfied directly, without first sending the messages to a topic host queue manager.

Note that, because of the critical role played by each topic host queue manager, you must choose queue managers that can handle the load, availability and connectivity requirements of topic hosting.

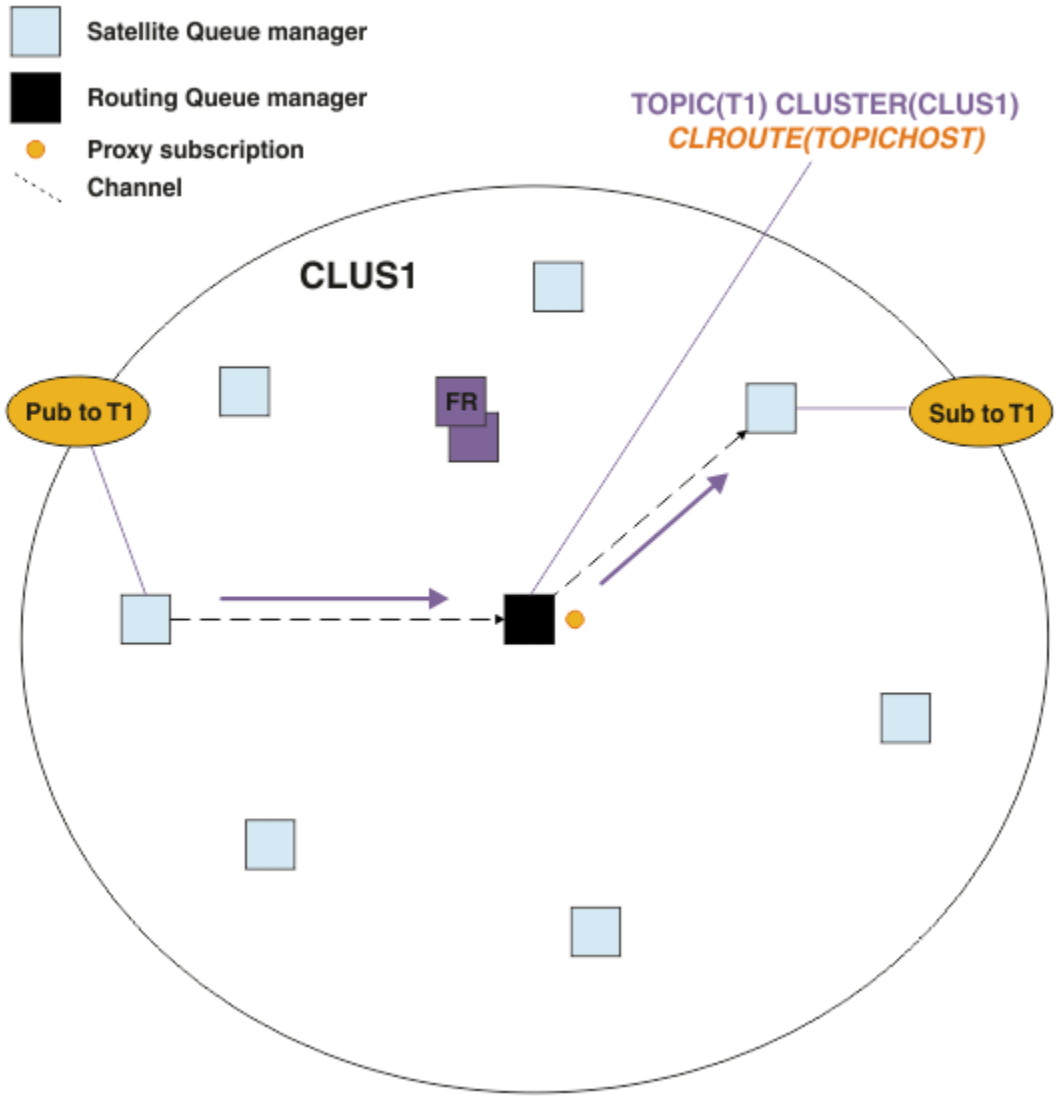


Figure 23. A topic host routed publish/subscribe cluster with one topic, one subscriber and one publisher

Dividing the topic tree across multiple queue managers

A routed topic hosting queue manager is only responsible for the subscription knowledge and publication messages that relate to the branch of the topic tree that its administered topic object is configured for. If different topics are used by different publish/subscribe applications in the cluster, you can configure different queue managers to host different clustered branches of the topic tree. This allows scaling by reducing the publication traffic, subscription knowledge and channels on each topic host queue manager in the cluster. You should use this method for distinct high volume branches of the topic tree:

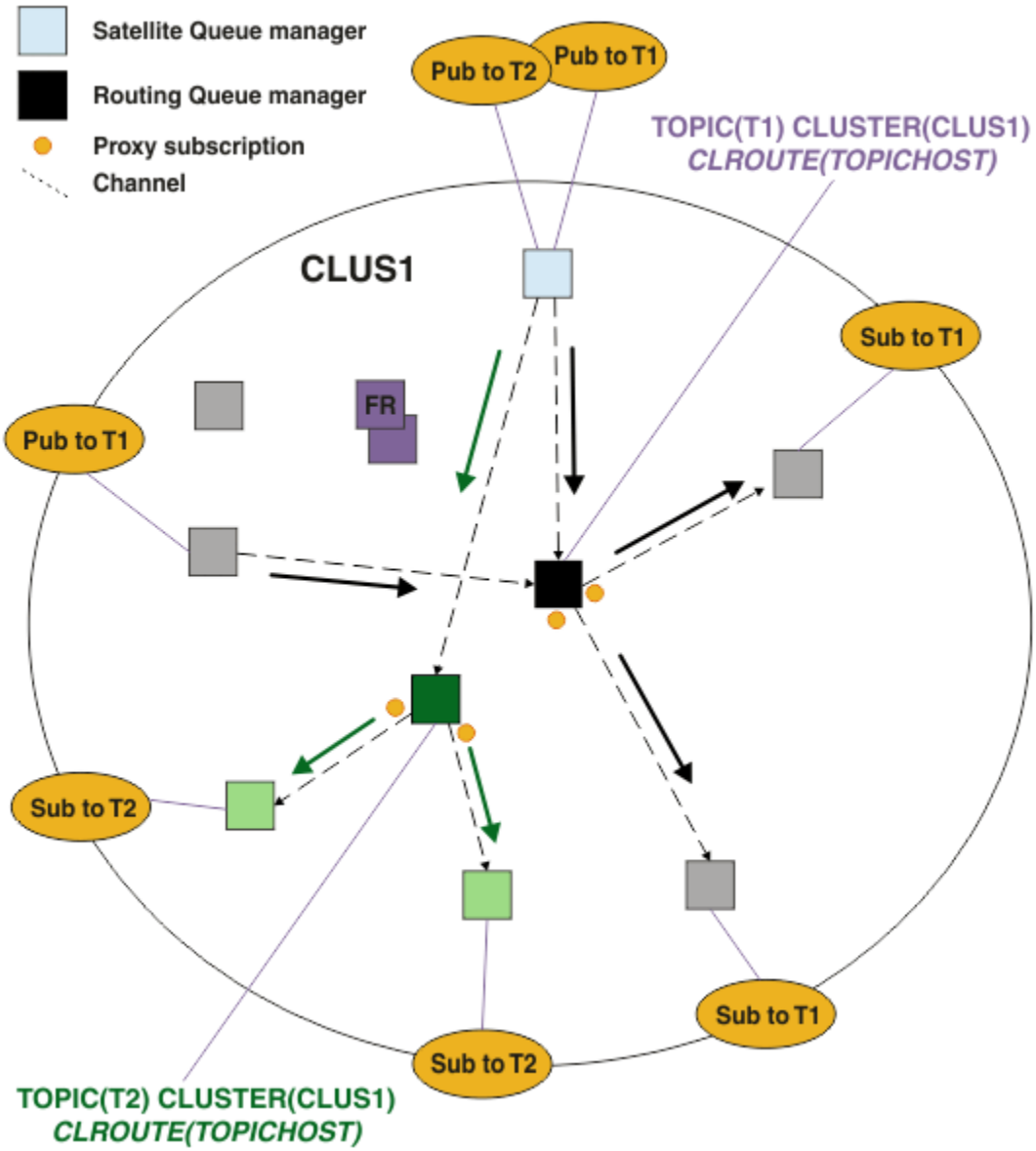


Figure 24. A topic host routed publish/subscribe cluster with two topics, each defined on one topic host

For example, using the topics described in Topic trees, if topic T1 was configured with a topic string of `/USA/Alabama`, and topic T2 was configured with a topic string of `/USA/Alaska`, then a message published to `/USA/Alabama/Mobile` would be routed through the queue manager hosting T1, and a message published to `/USA/Alaska/Juneau` would be routed through the queue manager hosting T2.

Note: You cannot make a single subscription span multiple clustered branches of the topic tree by using a wildcard higher in the topic tree than the points that are clustered. See Wildcard subscriptions.

Topic host routing using multiple topic hosts for a single topic

If a single queue manager has the responsibility for the routing of a topic, and that queue manager becomes unavailable or incapable of handling the workload, publications will not flow promptly to the subscriptions.

If you need greater resiliency, scalability and workload balancing than you get when you define a topic on just one queue manager, you can define a topic on more than one queue manager. Each individual message published is routed through a single topic host. When multiple matching topic host definitions exist, one of the topic hosts is chosen. The choice is made in the same way as for clustered queues. This allows messages to be routed to available topic hosts, avoiding any that are unavailable, and

allows message load to be workload balanced across multiple topic host queue managers and channels. However, ordering across multiple messages is not maintained when you use multiple topic hosts for the same topic in the cluster.

The following diagram shows a topic host routed cluster in which the same topic has been defined on two queue managers. In this example, the subscribing queue managers send information about the subscribed topic to both topic host queue managers in the form of a proxy subscription:

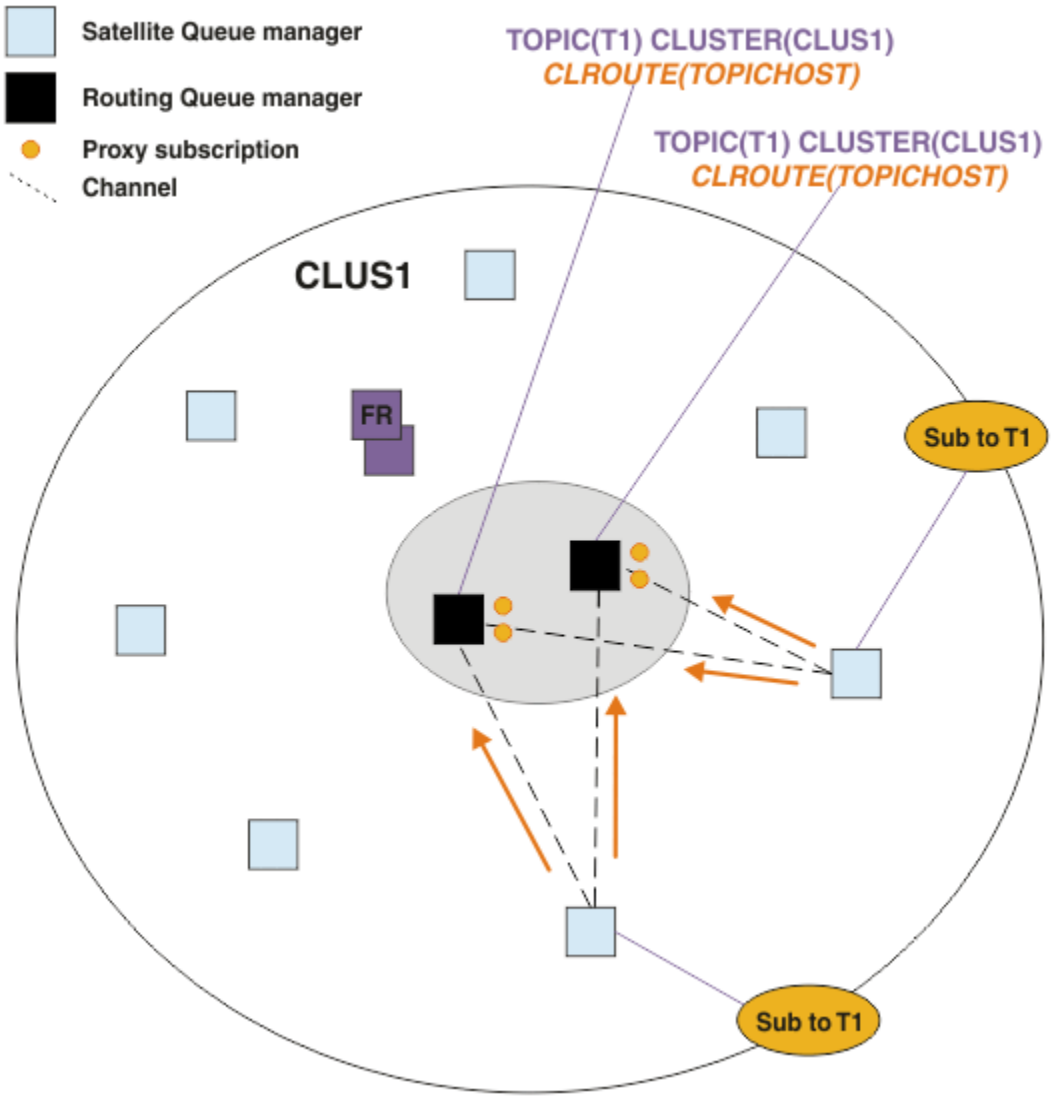


Figure 25. Creating proxy subscriptions in a multiple topic host publish/subscribe cluster

When a publication is made from a non-hosting queue manager, the queue manager sends a copy of the publication to *one* of the topic host queue managers for that topic. The system chooses the host based on the default behavior of the cluster workload management algorithm. In a typical system, this approximates to a round-robin distribution across each topic host queue manager. There is no affinity between messages from the same publishing application; this equates to using a cluster bind type of `NOTFIXED`.

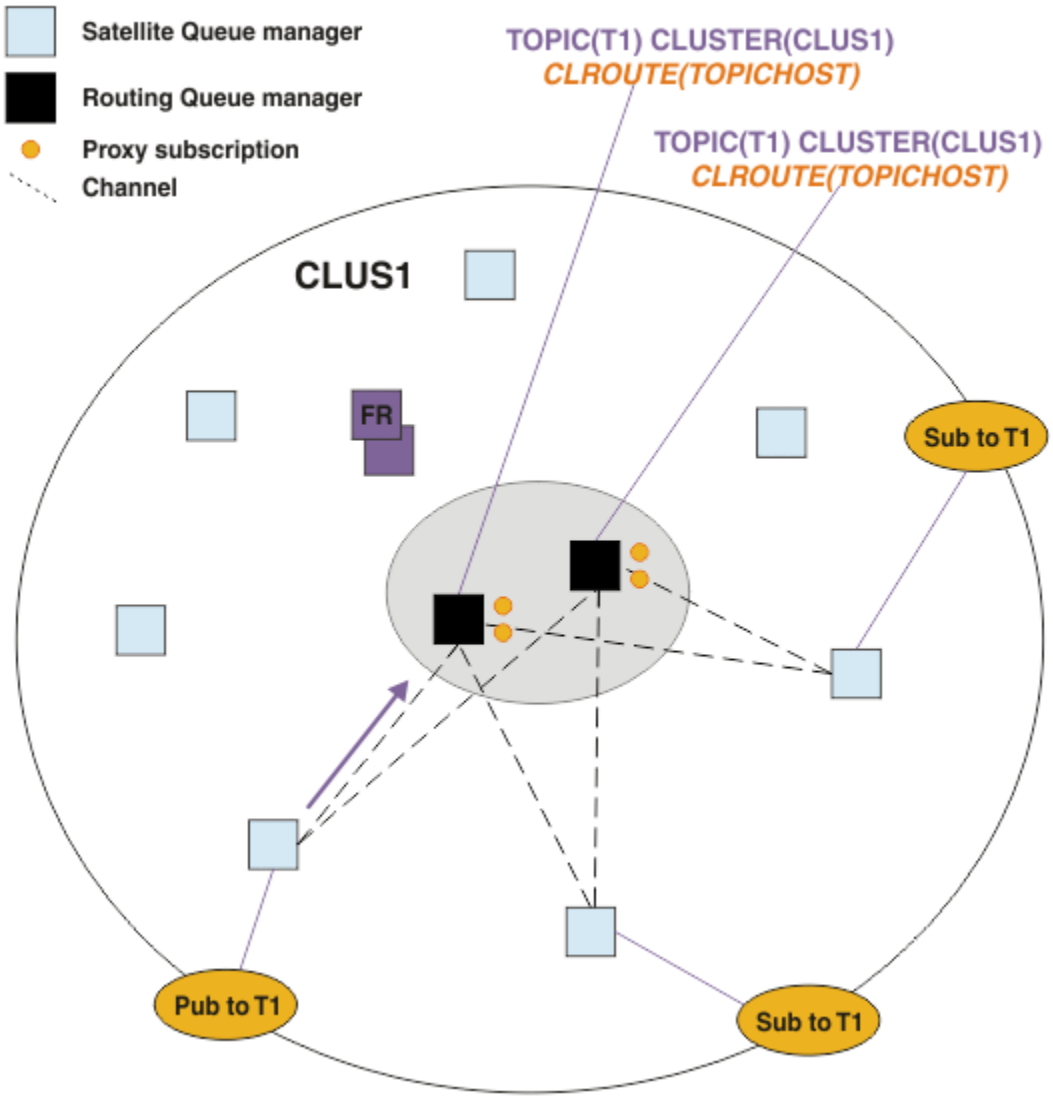


Figure 26. Receiving publications in a multiple topic host publish/subscribe cluster

Inbound publications to the chosen topic host queue manager are then forwarded to all queue managers that have registered a matching proxy subscription:

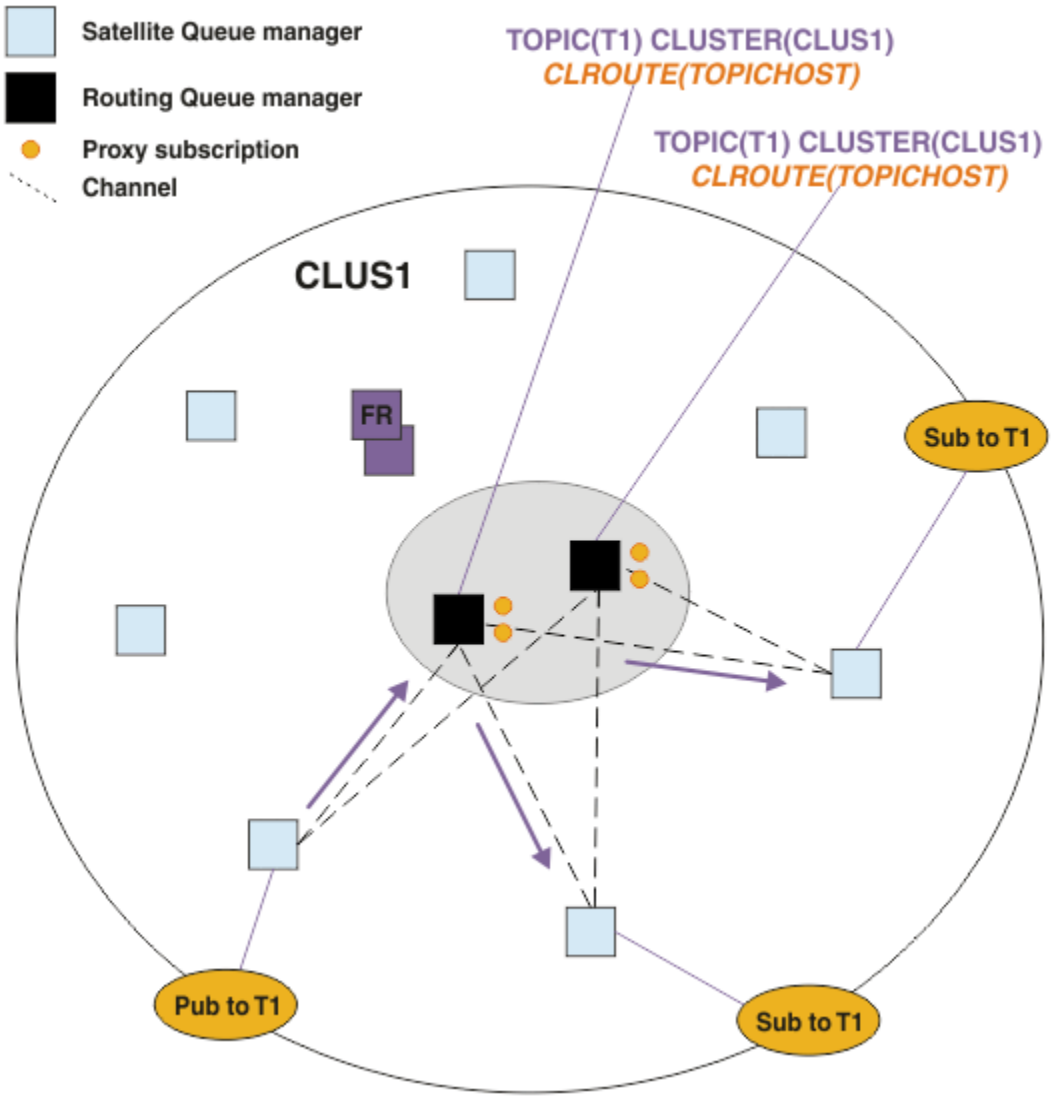


Figure 27. Routing publications to subscribers in a multiple topic host publish/subscribe cluster

Making subscriptions and publishers local to a topic host queue manager

The above examples show the routing between publishers and subscribers on queue managers that do not host administered routed topic objects. In these topologies, messages require multiple *hops* to reach the subscriptions.

Where the additional hop is not desirable, it might be appropriate to connect key publishers to topic hosting queue managers. However, if there are multiple topic hosts for a topic and only one publisher, all publication traffic will be routed through the topic host queue manager that the publisher is connected to.

Similarly, if there are key subscriptions, these could be located on a topic host queue manager. However, if there are multiple hosts of the routed topic, only a proportion of the publications will avoid the additional hop, with the remainder being routed through the other topic host queue managers first.

Topologies such as these are described further here: Topic host routing using centralized publishers or subscribers.

Note: Special planning is needed if changing the routed topic configuration when co-locating publishers or subscriptions with routed topic hosts. For example, see Adding extra topic hosts to a topic host routed cluster.

Summary and additional considerations

A topic host routed publish/subscribe cluster gives you precise control over which queue managers host each topic, and those queue managers become the *routing* queue managers for that branch of the topic tree. Moreover, queue managers without subscriptions or publishers have no need to connect with the topic host queue managers, and queue managers with subscriptions have no need to connect to queue managers that do not host a topic. This configuration can significantly reduce the number of connections between queue managers in the cluster, and the amount of information being passed between queue managers. This is especially true in large clusters where only a subset of queue managers are performing publish/subscribe work. This configuration also gives you some control over the load on individual queue managers in the cluster, so (for example) you can choose to host highly active topics on more powerful and more resilient systems. For certain configurations - notably larger clusters - it is usually a more appropriate topology than *direct routing*.

However, topic host routing also imposes certain constraints upon your system:

- System configuration and maintenance require more planning than for direct routing. You need to decide which points to cluster in the topic tree, and the location of the topic definitions in the cluster.
- Just as for direct routed topics, when a new topic host routed topic is defined, the information is pushed to the full repository queue managers, and from there direct to all members of the cluster. This event causes channels to be started to each member of the cluster from the full repositories if not already started.
- Publications are always sent to a host queue manager from a non-host queue manager, even if there are no subscriptions in the cluster. Therefore, you should use routed topics when subscriptions are typically expected to exist, or when the overhead of global connectivity and knowledge is greater than the risk of extra publication traffic.

  Note: As previously described, making publishers local to a topic host can mitigate this risk.
- Messages that are published on non-host queue managers do not go direct to the queue manager that hosts the subscription, they are always routed through a topic host queue manager. This approach can increase the total overhead to the cluster, and increase message latency and reduce performance.

  Note: As previously described, making subscriptions or publishers local to a topic host can mitigate this risk.
- Using a single topic host queue manager introduces a single point of failure for all messages that are published to a topic. You can remove this single point of failure by defining multiple topic hosts. However, having multiple hosts affects the order of published messages as received by subscriptions.
- Extra message load is incurred by topic host queue managers, because publication traffic from multiple queue managers needs to be processed by them. This load can be lessened: Either use multiple topic hosts for a single topic (in which case message ordering is not maintained), or use different queue managers to host routed topics for different branches of the topic tree.

Before you use topic host routing, explore the alternative approaches detailed in "Direct routing in publish/subscribe clusters" on page 73, and "Routing in publish/subscribe hierarchies" on page 102.

Publish/subscribe clustering: Best practices

Using clustered topics makes extending the publish/subscribe domain between queue managers simple, but can lead to problems if the mechanics and implications are not fully understood. There are two models for information sharing and publication routing. Implement the model that best meets your individual business needs, and performs best on your chosen cluster.

The best practice information in the following sections does not provide a one size fits all solution, but rather shares common approaches to solving common problems. It assumes that you have a basic understanding of IBM MQ clusters, and of publish/subscribe messaging, and that you are familiar with the information in Distributed publish/subscribe networks and "Designing publish/subscribe clusters" on page 71.

When you use a cluster for point-to-point messaging, each queue manager in the cluster works on a need-to-know basis. That is, it only finds out about other cluster resources, such as other queue managers in the cluster and clustered queues, when applications connecting to them request to use them. When

you add publish/subscribe messaging to a cluster, an increased level of sharing of information and connectivity between cluster queue managers is introduced. To be able to follow best practices for publish/subscribe clusters, you need to fully understand the implications of this change in behavior.

To allow you to build the best architecture, based on your precise needs, there are two models for information sharing and publication routing in publish/subscribe clusters: *direct routing* and *topic host routing*. To make the right choice, you need to understand both models, and the different requirements that each model satisfies. These requirements are discussed in the following sections, in conjunction with "Planning your distributed publish/subscribe network" on page 68:

- "Reasons to limit the number of cluster queue managers involved in publish/subscribe activity" on page 88
- "How to decide which topics to cluster" on page 88
- "How to size your system" on page 89
- "Publisher and subscription location" on page 90
- "Publication traffic" on page 90
- "Subscription change and dynamic topic strings" on page 91

Reasons to limit the number of cluster queue managers involved in publish/subscribe activity

There are capacity and performance considerations when you use publish/subscribe messaging in a cluster. Therefore, it is best practice to consider carefully the need for publish/subscribe activity across queue managers, and to limit it to only the number of queue managers that require it. After the minimum set of queue managers that need to publish and subscribe to topics are identified, they can be made members of a cluster that contains only them and no other queue managers.

This approach is especially useful if you have an established cluster already functioning well for point-to-point messaging. When you are turning an existing large cluster into a publish/subscribe cluster, it is a better practice to initially create a separate cluster for the publish/subscribe work where the applications can be tried, rather than using the current cluster. You can use a subset of existing queue managers that are already in one or more point-to-point clusters, and make this subset members of the new publish/subscribe cluster. However, the full repository queue managers for your new cluster must not be members of any other cluster; this isolates the additional load from the existing cluster full repositories.

If you cannot create a new cluster, and have to turn an existing large cluster into a publish/subscribe cluster, do not use a direct routed model. The topic host routed model usually performs better in larger clusters, because it generally restricts the publish/subscribe information sharing and connectivity to the set of queue managers that are actively performing publish/subscribe work, concentrating on the queue managers hosting the topics. The exception to that is if a manual refresh of the subscription information is invoked on a queue manager hosting a topic definition, at which point the topic host queue manager will connect to every queue manager in the cluster. See Resynchronization of proxy subscriptions.

If you establish that a cluster cannot be used for publish/subscribe due to its size or current load, it is good practice to prevent this cluster unexpectedly being made into a publish/subscribe cluster. Use the **PSCLUS** queue manager property to stop anyone adding a clustered topic on any queue manager in the cluster. See "Inhibiting clustered publish/subscribe" on page 97.

How to decide which topics to cluster

It is important to choose carefully which topics are added to the cluster: The higher up the topic tree these topics are, the more widespread their use becomes. This can result in more subscription information and publications being propagated than necessary. If there are multiple, distinct branches of the topic tree, where some need to be clustered and some do not, create administered topic objects at the root of each branch that needs clustering and add those to the cluster. For example, if branches `/A`, `/B` and `/C` need clustering, define a separate clustered topic objects for each branch.

Note: The system prevents you from nesting clustered topic definitions in the topic tree. You are only permitted to cluster topics at one point in the topic tree for each sub branch. For example, you cannot

define clustered topic objects for /A and for /A/B. Nesting clustered topics can lead to confusion over which clustered object applies to which subscription, especially when subscriptions are using wildcards. This is even more important when using topic host routing, where routing decisions are precisely defined by your allocation of topic hosts.

If clustered topics must be added high up the topic tree, but some branches of the tree below the clustered point do not require the clustered behavior, you can use the subscription and publication scope attributes to reduce the level of subscription and publication sharing for further topics.

You should not put the topic root node into the cluster without considering the behavior that is seen. Make global topics obvious where possible, for example by using a high-level qualifier in the topic string: `/global` or `/cluster`.

There is a further reason for not wanting to make the root topic node clustered. This is because every queue manager has a local definition for the root node, the `SYSTEM.BASE.TOPIC` topic object. When this object is clustered on one queue manager in the cluster, all other queue managers are made aware of it. However, when a local definition of the same object exists, its properties override the cluster object. This results in those queue managers acting as if the topic was not clustered. To resolve this, you would need to cluster every definition of `SYSTEM.BASE.TOPIC`. You could do this for direct routed definitions, but not for topic host routed definitions, because it causes every queue manager to become a topic host.

How to size your system

Publish/subscribe clusters typically result in a different pattern of cluster channels to point-to-point messaging in a cluster. The point-to-point model is an 'opt in' one, but publish/subscribe clusters have a more indiscriminate nature with subscription fan-out, especially when using direct routed topics. Therefore, it is important to identify which queue managers in a publish/subscribe cluster will use cluster channels to connect to other queue managers, and under what circumstances.

The following table lists the typical set of cluster sender and receiver channels expected for each queue manager in a publish/subscribe cluster under normal running, dependent on the queue manager role in the publish/subscribe cluster.

Table 5. Cluster sender and receiver channels for each routing method.

Queue manager role	Direct cluster receivers	Direct cluster senders	Topic cluster receivers	Topic cluster senders
Full repository	AllQmgrs	AllQmgrs	AllQmgrs	AllQMgrs
Host of topic definition	n/a	n/a	AllSubs+AllPubs (1)	AllSubs (1)
Subscriptions created	AllPubs (1)	AllQMgrs	AllHosts	AllHosts
Publishers connected	AllSubs (1)	AllSubs (1)	AllHosts	AllHosts
No publishers or subscribers	AllSubs (1)	None (1)	None (2)	None (2)

Key:

AllQmgrs
A channel to and from every queue manager in the cluster.

AllSubs
A channel to and from every queue manager where a subscription has been created.

AllPubs
A channel to and from every queue manager where a publishing application has been connected.

AllHosts
A channel to and from every queue manager where a definition of the clustered topic object has been configured.

None
No channels to or from other queue managers in the cluster for the sole purpose of publish/subscribe messaging.

Notes:

1. If a queue manager refresh of proxy subscriptions is made from this queue manager, a channel to and from all other queue managers in the cluster might be automatically created.
2. If a queue manager refresh of proxy subscriptions is made from this queue manager, a channel to and from any other queue managers in the cluster that host a definition of a clustered topic might be automatically created.

The previous table shows that topic host routing typically uses significantly less cluster sender and receiver channels than direct routing. If channel connectivity is a concern for certain queue managers in a cluster, for reasons of capacity or ability to establish certain channels (for example, through firewalls), topic host routing is therefore a preferred solution.

Publisher and subscription location

Clustered publish/subscribe enables messages published on one queue manager to be delivered to subscriptions on any other queue manager in the cluster. As for point-to-point messaging, the cost of transmitting messages between queue managers can be detrimental to performance. Therefore you should consider creating subscriptions to topics on the same queue managers as where messages are being published.

When using topic host routing within a cluster, it is important to also consider the location of the subscriptions and publishers with respect to the topic hosting queue managers. When the publisher is not connected to a queue manager that is a host of the clustered topic, messages published are always sent to a topic hosting queue manager. Similarly, when a subscription is created on a queue manager that is not a topic host for a clustered topic, messages published from other queue managers in the cluster are always sent to a topic hosting queue manager first. More specifically, if the subscription is located on a queue manager that hosts the topic, but there is one or more other queue managers that also host that same topic, a proportion of publications from other queue managers are routed through those other topic hosting queue managers. See Topic host routing using centralized publishers or subscribers for more information on designing a topic host routed publish/subscribe cluster to minimize the distance between publishers and subscriptions.

Publication traffic

Messages published by an application connected to one queue manager in a cluster are transmitted to subscriptions on other queue managers using cluster sender channels.

When you use direct routing, the messages published take the shortest path between queue managers. That is, they go direct from the publishing queue manager to each of the queue managers with subscriptions. Messages are not transmitted to queue managers that do not have subscriptions for the topic. See Proxy subscriptions in a publish/subscribe network.

Where the rate of publication messages between any one queue manager and another in the cluster is high, the cluster channel infrastructure between those two points must be able to maintain the rate. This might involve tuning the channels and transmission queue being used.

When you use topic host routing, each message published on a queue manager that is not a topic host is transmitted to a topic host queue manager. This is independent of whether one or more subscriptions exist anywhere else in the cluster. This introduces further factors to consider in planning:

- Is the additional latency of first sending each publication to a topic host queue manager acceptable?
- Can each topic host queue manager sustain the inbound and outbound publication rate? Consider a system with publishers on many different queue managers. If they all send their messages to a very

small set of topic hosting queue managers, those topic hosts might become a bottleneck in processing those messages and routing them on to subscribing queue managers.

- Is it expected that a significant proportion of the published messages will not have a matching subscriber? If so, and the rate of publishing such messages is high, it might be best to make the publisher's queue manager a topic host. In that situation, any published message where no subscriptions exist in the cluster will not be transmitted to any other queue managers.

These problems might also be eased by introducing multiple topic hosts, to spread the publication load across them:

- Where there are multiple distinct topics, each with a proportion of the publication traffic, consider hosting them on different queue managers.
- If the topics cannot be separated onto different topic hosts, consider defining the same topic object on multiple queue managers. This results in publications being workload balanced across each of them for routing. However, this is only appropriate when publication message ordering is not required.

Subscription change and dynamic topic strings

Another consideration is the effect on performance of the system for propagating proxy subscriptions. Typically, a queue manager sends a proxy subscription message to certain other queue managers in the cluster when the first subscription for a specific clustered topic string (not just a configured topic object) is created on that queue manager. Similarly, a proxy subscription deletion message is sent when the last subscription for a specific clustered topic string is deleted.

For direct routing, each queue manager with subscriptions sends those proxy subscriptions to every other queue manager in the cluster. For topic host routing, each queue manager with subscriptions only sends the proxy subscriptions to each queue manager that hosts a definition for that clustered topic. Therefore, with direct routing, the more queue managers there are in the cluster, the higher the overhead of maintaining proxy subscriptions across them. Whereas, with topic host routing, the number of queue managers in the cluster is not a factor.

In both routing models, if a publish/subscribe solution consists of many unique topic strings being subscribed to, or the topics on a queue manager in the cluster are frequently being subscribed and unsubscribed, a significant overhead will be seen on that queue manager, caused by constantly generating messages distributing and deleting the proxy subscriptions. With direct routing, this is compounded by the need to send these messages to every queue manager in the cluster.

If the rate of change of subscriptions is too high to accommodate, even within a topic host routed system, see Subscription performance in publish/subscribe networks for information about ways to reduce proxy subscription overhead.

Defining cluster topics

Cluster topics are administrative topics with the **cluster** attribute defined. Information about cluster topics is pushed to all members of a cluster, and combined with local topics to create portions of a topic space that spans multiple queue managers. This enables messages published on a topic on one queue manager to be delivered to subscriptions of other queue managers in the cluster.

When you define a cluster topic on a queue manager, the cluster topic definition is sent to the full repository queue managers. The full repositories then propagate the cluster topic definition to all queue managers within the cluster, making the same cluster topic available to publishers and subscribers at any queue manager in the cluster. The queue manager on which you create a cluster topic is known as a cluster topic host. The cluster topic can be used by any queue manager in the cluster, but any modifications to a cluster topic must be made on the queue manager where that topic is defined (the host) at which point the modification is propagated to all members of the cluster through the full repositories.

When you use direct routing, the location of the clustered topic definition does not directly affect the behavior of the system, because all queue managers in the cluster use the topic definition in the same way. You should therefore define the topic on any queue manager that will be a member of the cluster for

as long as the topic is needed, and that is on a system reliable enough to regularly be in contact with the full repository queue managers.

When you use topic host routing, the location of the clustered topic definition is very important, because other queue managers in the cluster create channels to this queue manager and send subscription information and publications to it. To choose the best queue manager to host the topic definition, you need to understand topic host routing. See "Topic host routing in publish/subscribe clusters" on page 78.

If you have a clustered topic, and a local topic object, then the local topic takes precedence. See "Multiple cluster topic definitions of the same name" on page 94.

For information about the commands to use to display cluster topics, see the related information.

Clustered topic inheritance

Typically, publishing and subscribing applications in a clustered publish/subscribe topology expect to work the same, no matter which queue manager in the cluster they are connected to. This is why clustered administered topic objects are propagated to every queue manager in the cluster.

An administered topic object inherits its behavior from other administered topic objects higher in the topic tree. This inheritance occurs when an explicit value has not been set for a topic parameter.

In the case of clustered publish/subscribe, it is important to consider such inheritance because it introduces the possibility that publishers and subscribers will behave differently depending on which queue manager they connect to. If a clustered topic object leaves any parameters to inherit from higher topic objects, the topic might behave differently on different queue managers in the cluster. Similarly, locally defined topic objects defined below a clustered topic object in the topic tree will mean those lower topics are still clustered, but the local object might change its behavior in some way that differs from other queue managers in the cluster.

Wildcard subscriptions

Proxy subscriptions are created when local subscriptions are made to a topic string that resolves at, or below, a clustered topic object. If a wildcard subscription is made higher in the topic hierarchy than any cluster topic, it does not have proxy subscriptions sent around the cluster for the matching cluster topic, and therefore receives no publications from other members of the cluster. It does however receive publications from the local queue manager.

However, if another application subscribes to a topic string that resolves to or below the cluster topic, proxy subscriptions are generated and publications are propagated to this queue manager. On arrival the original, higher wildcard subscription is considered a legitimate recipient of those publications and receives a copy. If this behavior is not required, set **`WILDCARD(BLOCK)`** on the clustered topic. This makes the original wildcard not be considered a legitimate subscription, and stops it receiving any publications (local, or from elsewhere in the cluster) on the cluster topic, or its subtopics.

Related concepts

Working with administrative topics

Working with subscriptions

Related reference

DISPLAYTOPIC

DISPLAYTPSTATUS

DISPLAYSUB

Cluster topic attributes

When a topic object has the cluster name attribute set, the topic definition is propagated across all queue managers in the cluster. Each queue manager uses the propagated topic attributes to control the behavior of publish/subscribe applications.

A topic object has a number of attributes that apply to publish/subscribe clusters. Some control the general behavior of the publishing and subscribing applications and some control how the topic is used across the cluster.

A clustered topic object definition must be configured in a way that all queue managers in the cluster can correctly use it.

For example if the model queues to be used for managed subscriptions (MDURMDL and MNDURMDL) are set to a non-default queue name, that named model queue must be defined on all queue managers where managed subscriptions will be created.

Similarly, if any attribute is set to ASPARENT, the behavior of the topic will be dependent on the higher nodes in the topic tree (see Administrative topic objects) on each individual queue manager in the cluster. This might result in different behavior when publishing or subscribing from different queue managers.

The main attributes that directly relate to publish/subscribe behavior across the cluster are as follows:

CLROUTE
This parameter controls the routing of messages between queue managers where publishers are connected, and queue managers where matching subscriptions exist.

- You configure the route to be either direct between these queue managers, or through a queue manager that hosts a definition of the clustered topic. See Publish/subscribe clusters for more details.
- You cannot change the **CLROUTE** while the **CLUSTER** parameter is set. To change the **CLROUTE**, first set the **CLUSTER** property to be blank. This stops applications that use the topic from behaving in a clustered manner. This in turn results in a break in publications being delivered to subscriptions, so you should also quiesce publish/subscribe messaging while making the change.

PROXYSUB
This parameter controls when proxy subscriptions are made.

- FIRSTUSE is the default value, and causes proxy subscriptions to be sent in response to local subscriptions on a queue manager in a distributed publish/subscribe topology, and canceled when no longer required. For details about why you might want to change this attribute from the default value of FIRSTUSE, see Individual proxy subscription forwarding and *publish everywhere* .
- To enable *publish everywhere*, you set the **PROXYSUB** parameter to FORCE for a high-level topic object. This results in a single wildcard proxy subscription that matches all topics below this topic object in the topic tree.

Note: Setting the **PROXYSUB(FORCE)** attribute in a large or busy publish/subscribe cluster can result in excessive load on system resources. The **PROXYSUB(FORCE)** attribute is propagated to every queue manager, not just the queue manager that the topic was defined on. This causes every queue manager in the cluster to create a wildcarded proxy subscription.

A copy of a message to this topic, published on any queue manager in the cluster, is sent to every queue manager in the cluster - either directly, or through a topic host queue manager, depending on the **CLROUTE** setting.

When the topic is direct routed, every queue manager creates cluster sender channels to every other queue manager. When the topic is topic host routed, channels to each topic host queue manager are created from every queue manager in the cluster.

For more information about the **PROXYSUB** parameter when used in clusters, see Direct routed publish/subscribe performance.

PUBSCOBE and SUBSCOPE
These parameters determine whether this queue manager propagates publications to queue managers in the topology (publish/subscribe cluster or hierarchy) or restricts the scope to just its local queue manager. You can do the equivalent job programmatically using MQPMO_SCOPE_QMGR and MQSO_SCOPE_QMGR.

PUBSCOPE
If a cluster topic object is defined with **PUBSCOPE(QMGR)**, the definition is shared with the cluster, but the scope of publications that are based on that topic is local only and they are not sent to other queue managers in the cluster.

SUBSCOPE
If a cluster topic object is defined with **SUBSCOPE(QMGR)**, the definition is shared with the cluster, but the scope of subscriptions that are based on that topic is local only, therefore no proxy subscriptions are sent to other queue managers in the cluster.

These two attributes are commonly used together to isolate a queue manager from interacting with other members of the cluster on particular topics. The queue manager neither publishes or receives publications on those topics to and from other members of the cluster. This situation does not prevent publication or subscription if topic objects are defined on subtopics.

Setting **SUBSCOPE** to QMGR on a local definition of a topic does not prevent other queue managers in the cluster from propagating their proxy subscriptions to the queue manager if they are using a clustered version of the topic, with **SUBSCOPE(ALL)**. However, if the local definition also sets **PUBSCOPE** to QMGR those proxy subscriptions are not sent publications from this queue manager.

Related concepts
Publication scope
Subscription scope

Multiple cluster topic definitions of the same name
You can define the same named cluster topic object on more than one queue manager in the cluster, and in certain scenarios this enables specific behavior. When multiple cluster topic definitions exist of the same name, the majority of properties should match. If they do not, errors or warnings are reported depending on the significance of the mismatch.

In general, if there is a mismatch in the properties of multiple cluster topic definitions, warnings are issued and one of the topic object definitions is used by each queue manager in the cluster. Which definition is used by each queue manager is not deterministic, or consistent across the queue managers in the cluster. Such mismatches should be resolved as quickly as possible.

During cluster setup or maintenance you sometimes need to create multiple cluster topic definitions that are not identical. However this is only ever useful as a temporary measure, and it is therefore treated as a potential error condition.

When mismatches are detected, the following warning messages are written to each queue manager's error log:

- Multi On Multiplatforms, AMQ9465 and AMQ9466.
- z/OS On z/OS, CSQX465I and CSQX466I.

The chosen properties for any topic string on each queue manager can be determined by viewing topic status rather than the topic object definitions, for example by using **DISPLAY TPSTATUS**.

In some situations, a conflict in configuration properties is severe enough to stop the topic object being created, or to cause the mismatched objects to be marked as invalid and not propagated across the cluster (See **CLSTATE** in DISPLAY TOPIC). These situations occur when there is a conflict in the cluster routing property (**CLROUTE**) of the topic definitions. Additionally, due to the importance of consistency across topic host routed definitions, further inconsistencies are rejected as detailed in subsequent sections of this article.

If the conflict is detected at the time that the object is defined, the configuration change is rejected. If detected later by the full repository queue managers, the following warning messages are written to the queue managers error logs:

- Multi On Multiplatforms: AMQ9879
- z/OS On z/OS: CSQX879E.

When multiple definitions of the same topic object are defined in the cluster, a locally defined definition takes precedence over any remotely defined one. Therefore, if any differences exist in the definitions, the queue managers hosting the multiple definitions behave differently from each other.

The effect of defining a non-cluster topic with the same name as a cluster topic from another queue manager

It is possible to define an administered topic object that is not clustered on a queue manager that is in a cluster, and simultaneously define the same named topic object as a clustered topic definition on a different queue manager. In this case, the locally defined topic object takes precedence over all remote definitions of the same name.

This has the effect of preventing the clustering behavior of the topic when used from this queue manager. That is, subscriptions might not receive publications from remote publishers, and messages from publishers might not be propagated to remote subscriptions in the cluster.

Careful consideration should be given before configuring such a system, because this can lead to confusing behavior.

Note: If an individual queue manager needs to prevent publications and subscriptions from propagating around the cluster, even when the topic has been clustered elsewhere, an alternative approach is to set the publication and subscription scopes to only the local queue manager. See "Cluster topic attributes" on page 92.

Multiple cluster topic definitions in a direct routed cluster

For direct routing, you do not usually define the same cluster topic on more than one cluster queue manager. This is because direct routing makes the topic available at all queue managers in the cluster, no matter which queue manager it was defined on. Moreover, adding multiple cluster topic definitions significantly increases system activity and administrative complexity, and with increased complexity comes a greater chance of human error:

- Each definition results in an additional cluster topic object being pushed out to the other queue managers in the cluster, including the other cluster topic host queue managers.
- All definitions for a specific topic in a cluster must be identical, otherwise it is difficult to work out which topic definition is being used by a queue manager.

It is also not essential that the sole host queue manager is continually available for the topic to function correctly across the cluster, because the cluster topic definition is cached by the full repository queue managers and by all other queue managers in their partial cluster repositories. For more information, see Availability of topic host queue managers that use direct routing.

For a situation in which you might need to temporarily define a cluster topic on a second queue manager, for example when the existing host of the topic is to be removed from the cluster, see Moving a cluster topic definition to a different queue manager.

If you need to alter a cluster topic definition, take care to modify it at the same queue manager it was defined on. Attempting to modify it from another queue manager might accidentally create a second definition of the topic with conflicting topic attributes.

Multiple cluster topic definitions in a topic host routed cluster

When a cluster topic is defined with a cluster route of *topic host*, the topic is propagated across all queue managers in the cluster just as for *direct* routed topics. Additionally, all publish/subscribe messaging for that topic is routed through the queue managers where that topic is defined. Therefore the location and number of definitions of the topic in the cluster becomes important (see "Topic host routing in publish/subscribe clusters" on page 78).

To ensure adequate availability and scalability it is useful, if possible, to have multiple topic definitions. See Availability of topic host queue managers that use topic host routing.

When adding or removing additional definitions of a *topic host* routed topic in a cluster, you should consider the flow of messages at the time of the configuration change. If messages are being published in the cluster to the topic at the time of the change, a staged process is required to add or remove a topic definition. See Moving a cluster topic definition to a different queue manager and Adding extra topic hosts to a topic host routed cluster.

As previously explained, the properties of the multiple definitions should match, with the possible exception of the **PUB** parameter, as described in the next section. When publications are routed through topic host queue managers it is even more important for multiple definitions to be consistent. Therefore, an inconsistency detected in either the topic string or cluster name is rejected if one or more of the topic definitions has been configured for topic host cluster routing.

Note: Cluster topic definitions are also rejected if an attempt is made to configure them above or below another topic in the topic tree, where the existing clustered topic definition is configured for topic host routing. This prevents ambiguity in the routing of publications with respect to wildcarded subscriptions.

Special handling for the PUB parameter

The **PUB** parameter is used to control when applications can publish to a topic. In the case of topic host routing in a cluster, it can also control which topic host queue managers are used to route publications. For this reason it is permitted to have multiple definitions of the same topic object in the cluster, with different settings for the PUB parameter.

If multiple remote clustered definitions of a topic have different settings for this parameter, the topic allows publications to be sent and delivered to subscriptions if the following conditions are met:

- There is not a matching topic object defined on the queue manager that the publisher is connected to that is set to `PUB(DISABLED)`.
- One or more of the multiple topic definitions in the cluster is set to `PUB(ENABLED)`, or one or more of the multiple topic definitions is set to `PUB(ASPARENT)` and the local queue managers where the publisher is connected and the subscription defined are set to `PUB(ENABLED)` at a higher point in the topic tree.

For topic host routing, when messages are published by applications connected to queue managers that are not topic hosts, messages are only routed to the topic hosting queue managers where the **PUB** parameter has not explicitly been set to `DISABLED`. You can therefore use the `PUB(DISABLED)` setting to quiesce message traffic through certain topic hosts. You might want to do this to prepare for maintenance or removal of a queue manager, or for the reasons described in Adding extra topic hosts to a topic host routed cluster.

Availability of cluster topic host queue managers
Design your publish/subscribe cluster to minimize the risk that, should a topic host queue manager become unavailable, the cluster will no longer be able to process traffic for the topic. The effect of a topic host queue manager becoming unavailable depends on whether the cluster is using topic host routing or direct routing.

Availability of topic host queue managers that use direct routing

For direct routing, you do not usually define the same cluster topic on more than one cluster queue manager. This is because direct routing makes the topic available at all queue managers in the cluster, no matter which queue manager it was defined on.See Multiple cluster topic definitions in a direct routed cluster.

In a cluster, whenever the host of a clustered object (for example a clustered queue or clustered topic) becomes unavailable for a prolonged period of time, the other members of the cluster will eventually expire the knowledge of those objects. In the case of a clustered topic, if the cluster topic host queue manager becomes unavailable, the other queue managers continue to process publish/subscribe requests for the topic in a direct clustered way (that is, sending publications to subscriptions on remote queue managers) for at least 60 days from when the topic hosting queue manager was last in communication with the full repository queue managers. If the queue manager on which you defined the cluster topic object is never made available again, then eventually the cached topic objects on the other queue managers are deleted and the topic reverts to a local topic, in which case subscriptions cease to receive publications from applications connected to remote queue managers.

With the 60 day period to recover the queue manager on which you define a cluster topic object, there is little need to take special measures to guarantee that a cluster topic host remains available (note,

however, that any subscriptions defined on the unavailable cluster topic host do not remain available). The 60 day period is sufficient to cater for technical problems, and is likely to be exceeded only because of administrative errors. To mitigate that possibility, if the cluster topic host is unavailable, all members of the cluster write error log messages hourly, stating that their cached cluster topic object was not refreshed. Respond to these messages by making sure that the queue manager on which the cluster topic object is defined, is running. If it is not possible to make the cluster topic host queue manager available again, define the same clustered topic definition, with exactly the same attributes, on another queue manager in the cluster.

Availability of topic host queue managers that use topic host routing

For topic host routing, all publish/subscribe messaging for a topic is routed through the queue managers where that topic is defined. For this reason, it is very important to consider the continual availability of these queue managers in the cluster. If a topic host becomes unavailable, and no other host exists for the topic, traffic from publishers to subscribers on different queue managers in the cluster immediately halts for the topic. If additional topic hosts are available, the cluster queue managers route new publication traffic through these topic hosts, providing continuous availability of message routes.

As for direct topics, after 60 days, if the first topic host is still unavailable, knowledge of that topic host's topic is removed from the cluster. If this is the last remaining definition for this topic in the cluster, all other queue managers cease to forward publications to any topic host for routing.

To ensure adequate availability and scalability it is therefore useful, if possible, to define each topic on at least two cluster queue managers. This gives protection against any given topic host queue manager becoming unavailable. See also Multiple cluster topic definitions in a topic host routed cluster.

If you cannot configure multiple topic hosts (for example because you need to preserve message ordering), and you cannot configure just one topic host (because the availability of a single queue manager must not affect the flow of publications to subscriptions across all queue managers in the cluster), consider configuring the topic as a direct routed topic. This avoids reliance on a single queue manager for the whole cluster, but does still require each individual queue manager to be available in order for it to process locally hosted subscriptions and publishers.

Inhibiting clustered publish/subscribe

Introducing the first direct routed clustered topic into a cluster forces every queue manager in the cluster to become aware of every other queue manager, and potentially causes them to create channels to each other. If this is not desirable, you should instead configure topic host routed publish/subscribe. If the existence of a direct routed clustered topic might jeopardize the stability of the cluster, due to scaling concerns of each queue manager, you can completely disable clustered publish/subscribe functionality by setting **PSCLUS** to `DISABLED` on every queue manager in the cluster.

As described in“Direct routing in publish/subscribe clusters” on page 73, when you introduce a direct routed clustered topic into a cluster, all partial repositories are automatically notified of all other members of the cluster. The clustered topic might also create subscriptions at all other nodes (for example, where **PROXYSUB(FORCE)** is specified) and cause large numbers of channels to be started from a queue manager, even when there are no local subscriptions. This puts an immediate additional load on each queue manager in the cluster. For a cluster that contains many queue managers, this can cause a significant reduction in performance. Therefore the introduction of direct routed publish/subscribe to a cluster must be carefully planned.

When you know that a cluster cannot accommodate the overheads of direct routed publish/subscribe, you can instead use topic host routed publish/subscribe. For an overview of the differences, see “Designing publish/subscribe clusters” on page 71.

If you prefer to completely disable publish/subscribe functionality for the cluster, you can do so by setting the queue manager attribute **PSCLUS** to `DISABLED` on every queue manager in the cluster. This setting disables both direct routed and topic host routed publish/subscribe in the cluster, by modifying three aspects of queue manager functionality:

- An administrator of this queue manager is no longer able to define a `Topic` object as clustered.

- Incoming topic definitions or proxy subscriptions from other queue managers are rejected, and a warning message is logged to inform the administrator of incorrect configuration.
- Full repositories no longer automatically share information about every queue manager with all other partial repositories when they receive a topic definition.

Although **PSCLUS** is a parameter of each individual queue manager in a cluster, it is not intended to selectively disable publish/subscribe in a subset of queue managers in the cluster. If you selectively disable in this way, you will see frequent error messages. This is because proxy subscriptions and topic definitions are constantly seen and rejected if a topic is clustered on a queue manager where **PSCLUS** is enabled.

You should therefore aim to set **PSCLUS** to DISABLED on every queue manager in the cluster. However, in practice this state can be difficult to achieve and maintain, for example queue managers can join and leave the cluster at any time. At the very least, you must ensure that **PSCLUS** is set to DISABLED on all full repository queue managers. If you do this, and a clustered topic is subsequently defined on an ENABLED queue manager in the cluster, this does not cause the full repositories to inform every queue manager of every other queue manager, and so your cluster is protected from potential scaling issues across all queue managers. In this scenario, the origin of the clustered topic is reported in the error logs of the full repository queue managers.

If a queue manager participates in one or more publish/subscribe clusters, and also one or more point-to-point clusters, you must set **PSCLUS** to ENABLED on that queue manager. For this reason, when overlapping a point-to-point cluster with a publish subscribe cluster, you should use a separate set of full repositories in each cluster. This approach allows topic definitions and information about every queue manager to flow only in the publish/subscribe cluster.

To avoid inconsistent configurations when you change **PSCLUS** from ENABLED to DISABLED, no clustered topic objects can exist in any cluster of which this queue manager is a member. Any such topics, even remotely defined ones, must be deleted before changing **PSCLUS** to DISABLED.

For more information about **PSCLUS**, see ALTER QMGR (PSCLUS).

Related concepts

Direct routed publish/subscribe cluster performance

Publish/subscribe and multiple clusters

A single queue manager can be a member of more than one cluster. This arrangement is sometimes known as *overlapping clusters*. Through such an overlap, clustered queues can be made accessible from multiple clusters, and point-to-point message traffic can be routed from queue managers in one cluster to queue managers in another cluster. Clustered topics in publish/subscribe clusters do not provide the same capability. Therefore, their behavior must be clearly understood when using multiple clusters.

Unlike for a queue, you cannot associate a topic definition with more than one cluster. The scope of a clustered topic is confined to those queue managers in the same cluster as the topic is defined for. This allows publications to be propagated to subscriptions only on those queue managers in the same cluster.

A queue manager's topic tree

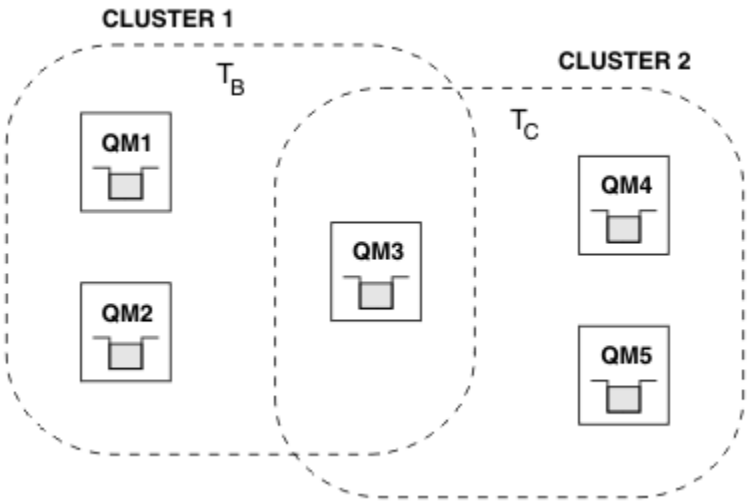


Figure 28. Overlapping clusters: Two clusters each subscribing to different topics

When a queue manager is a member of multiple clusters it is made aware of all clustered topics defined in each of those clusters. For example, in the previous figure QM3 is aware of both the T_B and T_C administered clustered topic objects, whereas QM1 is only aware of T_B. QM3 applies both topic definitions to its local topic, and therefore has a different behavor to QM1 for certain topics. For this reason it is important that clustered topics from different clusters do not interfere with each other. Interference can occur when one clustered topic is defined above or below another clustered topic in a different cluster (for example, they have topic strings of `/Sport` and `/Sport/Football`) or even for the same topic string in both. Another form of interference is when administered clustered topic objects are defined with the same object name in different clusters, but for different topic strings.

If such a configuration is made, the delivery of publications to matching subscriptions becomes very dependent on the relative locations of publishers and subscribers with respect to the cluster. For this reason, you cannot rely on such a configuration, and you should change it to remove the interfering topics.

When planning an overlapping cluster topology with publish/subscribe messaging, you can avoid any interference by treating the topic tree and clustered topic object names as if they span all overlapping clusters in the topology.

Integrating multiple publish/subscribe clusters

If there is a requirement for publish/subscribe messaging to span queue managers in different clusters, there are two options available:

- Connect the clusters together through the use of a publish/subscribe hierarchy configuration. See Combining the topic spaces of multiple clusters.
- Create an additional cluster that overlays the existing clusters and includes all queue managers that need to publish or subscribe to a particular topic.

With the latter option, you should consider carefully the size of the cluster and the most effective cluster routing mechanism. See“Designing publish/subscribe clusters” on page 71.

Design considerations for retained publications in publish/subscribe clusters

There are a few restrictions to consider when designing a publish/subscribe cluster to work with retained publications.

Considerations

Consideration 1: The following cluster queue managers always store the latest version of a retained publication:

- The publisher's queue manager
- In a topic host routed cluster, the topic host (provided there is only one topic host for the topic, as explained in the next section of this article)
- All queue managers with subscriptions matching the topic string of the retained publication

Consideration 2: Queue managers do not receive updated retained publications while they have no subscriptions. Therefore any retained publication stored on a queue manager that no longer subscribes to the topic will become stale.

Consideration 3: On creating any subscription, if there is a local copy of a retained publication for the topic string, the local copy is delivered to the subscription. If you are the first subscriber for any given topic string, a matching retained publication is also delivered from one of the following cluster members:

- In a direct routed cluster, the publisher's queue manager
- In a topic host routed cluster, the topic hosts for the given topic

Delivery of a retained publication from a topic host or publishing queue manager to the subscribing queue manager is asynchronous to the MQSUB calls. Therefore, if you use the MQSUBRQ call, the latest retained publication might be missed until a subsequent call to MQSUBRQ.

Implications

For any publish/subscribe cluster, when a first subscription is made, the local queue manager might be storing a stale copy of a retained publication and this is the copy that is delivered to the new subscription. The existence of a subscription on the local queue manager means that this will resolve the next time the retained publication is updated.

For a topic host routed publish/subscribe cluster, if you configure more than one topic host for a given topic, new subscribers might receive the latest retained publication from a topic host, or they might receive a stale retained publication from another topic host (with the latest having been lost). For topic host routing, it is usual to configure multiple topic hosts for a given topic. However, if you expect applications to make use of retained publications, you should configure only one topic host for each topic.

For any given topic string, you should use only a single publisher, and ensure the publisher always uses the same queue manager. If you do not do this, different retained publications might be active at different queue managers for the same topic, leading to unexpected behavior. Because multiple proxy subscriptions are distributed, multiple retained publications might be received.

If you are still concerned about subscribers using stale publications, consider setting a message expiry when you create each retained publication.

You can use the **CLEAR TOPICSTR** command to remove a retained publication from a publish/subscribe cluster. In certain circumstances you might need to issue the command on multiple members of the publish/subscribe cluster, as described in **CLEAR TOPICSTR**.

Wildcard subscriptions and retained publications

If you are using wildcard subscriptions, the corresponding proxy subscriptions delivered to other members of the publish/subscribe cluster are wildcarded from the topic separator immediately prior to the first wildcard character. See Wildcards and cluster topics.

Therefore the wildcard used might match more topic strings, and more retained publications, than will match the subscribing application.

This increases the amount of storage needed for the retained publications, and you therefore need to ensure that the hosting queue managers have enough storage capacity.

Related concepts

Retained publications

Individual proxy subscription forwarding and publish everywhere

REFRESH CLUSTER considerations for publish/subscribe clusters

Issuing the **REFRESH CLUSTER** command results in the queue manager temporarily discarding locally held information about a cluster, including any cluster topics and their associated proxy subscriptions.

The time taken from issuing the **REFRESH CLUSTER** command to the point that the queue manager regains a full knowledge of the necessary information for clustered publish/subscribe depends on the size of the cluster, the availability, and the responsiveness of the full repository queue managers.

During the refresh processing, disruption to publish/subscribe traffic in a publish/subscribe cluster occurs. For large clusters, use of the **REFRESH CLUSTER** command can disrupt the cluster while it is in progress, and again at 27 day intervals thereafter when the cluster objects automatically send status updates to all interested queue managers. See Refreshing in a large cluster can affect performance and availability of the cluster. For these reasons, the **REFRESH CLUSTER** command must be used in a publish/subscribe cluster only when under the guidance of IBM Support.

The disruption to the cluster can appear externally as the following symptoms:

- Subscriptions to cluster topics on this queue manager are not receiving publications from publishers that are connected to other queue managers in the cluster.
- Messages that are published to cluster topics on this queue manager are not being propagated to subscriptions on other queue managers.
- Subscriptions to cluster topics on this queue manager created during this period are not consistently sending proxy subscriptions to other members of the cluster.
- Subscriptions to cluster topics on this queue manager deleted during this period are not consistently removing proxy subscriptions from other members of the cluster.
- 10-second pauses, or longer, in message delivery.
- **MQPUT** failures, for example, MQRC_PUBLICATION_FAILURE.
- Publications placed on the dead-letter queue with a reason of MQRC_UNKNOWN_REMOTE_Q_MGR

For these reasons publish/subscribe applications need to be quiesced before issuing the **REFRESH CLUSTER** command.

After a **REFRESH CLUSTER** command is issued on a queue manager in a publish/subscribe cluster, wait until all cluster queue managers and cluster topics have been successfully refreshed, then resynchronize proxy subscriptions as described in Resynchronization of proxy subscriptions. When all proxy subscriptions have been correctly resynchronized, restart your publish/subscribe applications.

If a **REFRESH CLUSTER** command is taking a long time to complete, monitor it by looking at the CURDEPTH of `SYSTEM.CLUSTER.COMMAND.QUEUE`.

Related concepts

“Clustering: Using REFRESH CLUSTER best practices” on page 66

You use the **REFRESH CLUSTER** command to discard all locally held information about a cluster and rebuild that information from the full repositories in the cluster. You should not need to use this command, except in exceptional circumstances. If you do need to use it, there are special considerations about how you use it. This information is a guide based on testing and feedback from customers.

Related reference

Application issues seen when running REFRESH CLUSTER

MQSC Commands reference: REFRESH CLUSTER

Routing in publish/subscribe hierarchies

If your distributed queue manager topology is a publish/subscribe hierarchy, and a subscription is made on a queue manager, by default a proxy subscription is created on every queue manager in the hierarchy. Publications received on any queue manager are then routed through the hierarchy to each queue manager that hosts a matching subscription.

For an introduction to how messages are routed between queue managers in publish/subscribe hierarchies and clusters, see Distributed publish/subscribe networks.

When a subscription to a topic is made on a queue manager in a distributed publish/subscribe hierarchy, the queue manager manages the process by which the subscription is propagated to connected queue managers. *Proxy subscriptions* flow to all queue managers in the network. A proxy subscription gives a queue manager the information it needs to forward a publication to those queue managers that host subscriptions for that topic. Each queue manager in a publish/subscribe hierarchy is only aware of its direct relations. Publications put to one queue manager are sent, through its direct relations, to those queue managers with subscriptions. This is illustrated in the following figure, in which *Subscriber 1* registers a subscription for a particular topic on the *Asia* queue manager (1). Proxy subscriptions for this subscription on the *Asia* queue manager are forwarded to all other queue managers in the network (2,3,4).

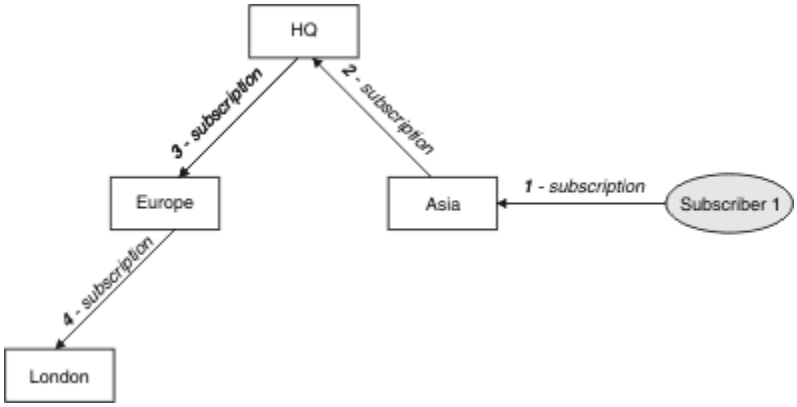


Figure 29. Propagation of subscriptions through a queue manager network

A queue manager consolidates all the subscriptions that are created on it, whether from local applications or from remote queue managers. It creates proxy subscriptions for the topics of the subscriptions with its neighbors, unless a proxy subscription already exists. This is illustrated in the following figure, in which *Subscriber 2* registers a subscription, to the same topic as in Figure 29 on page 102, on the *HQ* queue manager (5). The subscription for this topic is forwarded to the *Asia* queue manager, so that it is aware that subscriptions exist elsewhere on the network (6). The subscription is not forwarded to the *Europe* queue manager, because a subscription for this topic has already been registered; see step 3 in Figure 29 on page 102.

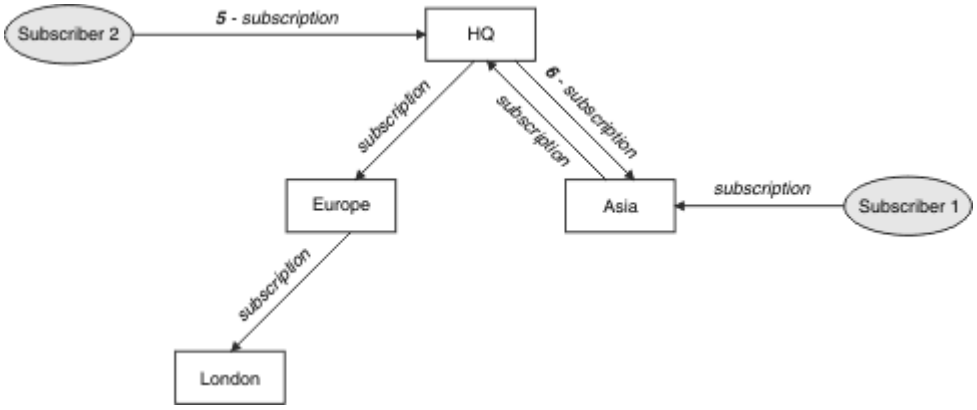


Figure 30. Multiple subscriptions

When an application publishes information to a topic, by default the receiving queue manager forwards it to all queue managers that have valid subscriptions to the topic. It might forward it through one or more intermediate queue managers. This is illustrated in the following figure, in which a publisher sends a publication, on the same topic as in Figure 30 on page 103, to the *Europe* queue manager (7). A subscription for this topic exists from *HQ* to *Europe*, so the publication is forwarded to the *HQ* queue manager (8). However, no subscription exists from *London* to *Europe* (only from *Europe* to *London*), so the publication is not forwarded to the *London* queue manager. The *HQ* queue manager sends the publication directly to *Subscriber 2* and to the *Asia* queue manager (9). The publication is forwarded to *Subscriber 1* from *Asia* (10).

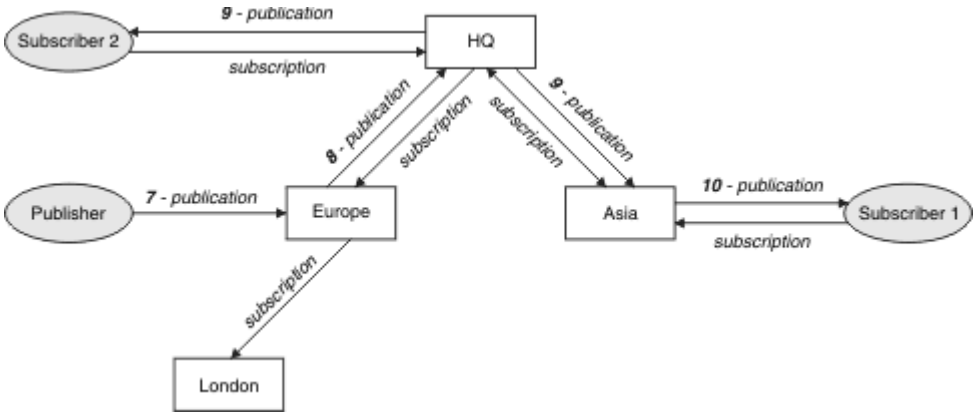


Figure 31. Propagation of publications through a queue manager network

When a queue manager sends any publications or subscriptions to another queue manager, it sets its own user ID in the message. If you are using a publish/subscribe hierarchy, and if the incoming channel is set up to put messages with the authority of the user ID in the message, then you must authorize the user ID of the sending queue manager. See Using default user IDs with a queue manager hierarchy.

Note: If you instead use publish/subscribe clusters, authorization is handled by the cluster.

Summary and additional considerations

A publish/subscribe hierarchy gives you precise control over the relationship between queue managers. After it has been created, it needs little manual intervention to administer. However it also imposes certain constraints upon your system:

- The higher nodes in the hierarchy, especially the root node, must be hosted on robust, highly available, and performant equipment. This is because more publication traffic is expected to flow through these nodes.
- The availability of every non-leaf queue manager in the hierarchy affects the ability of the network to flow messages from publishers to subscribers on other queue managers.
- By default, all topic strings subscribed to are propagated throughout the hierarchy, and publications are propagated only to remote queue managers that have a subscription to the associated topic. Therefore rapid changes to the set of subscriptions can become a limiting factor. You can change this default behavior, and instead have all publications propagated to all queue managers, which removes the need for proxy subscriptions. See Subscription performance in publish/subscribe networks.

  Note: A similar restriction also applies to direct routed clusters.
- Because of the interconnected nature of publish/subscribe queue managers, it takes time for proxy subscriptions to propagate around all nodes in the network. Remote publications do not necessarily start being subscribed to immediately, so early publications might not be sent following a subscription to a new topic string. You can remove the problems caused by the subscription delay by having all publications propagated to all queue managers, which removes the need for proxy subscriptions. See Subscription performance in publish/subscribe networks.

  Note: This restriction also applies to direct routed clusters.
- For a publish/subscribe hierarchy, adding or removing queue managers requires manual configuration to the hierarchy, with careful consideration to the location of those queue managers and their reliance on other queue managers. Unless you are adding or removing queue managers that are at the bottom of the hierarchy, and therefore have no further branches below them, you will also have to configure other queue managers in the hierarchy.

Before you use a publish/subscribe hierarchy as your routing mechanism, explore the alternative approaches detailed in “Direct routing in publish/subscribe clusters” on page 73 and “Topic host routing in publish/subscribe clusters” on page 78.

Distributed publish/subscribe system queues

Four system queues are used by queue managers for publish/subscribe messaging. You need to be aware of their existence only for problem determination and capacity planning purposes.

See Balancing producers and consumers in publish/subscribe networks for guidance on how to monitor these queues.

Table 6. Publish/subscribe system queues on Multiplatforms

System queue	Purpose
SYSTEM.INTER.QMGR.CONTROL	IBM MQ distributed publish/subscribe control queue
SYSTEM.INTER.QMGR.FANREQ	IBM MQ distributed publish/subscribe internal proxy subscription fan-out process input queue
SYSTEM.INTER.QMGR.PUBS	IBM MQ distributed publish/subscribe publications
SYSTEM.HIERARCHY.STATE	IBM MQ distributed publish/subscribe hierarchy relationship state

z/OS On z/OS, you set up the necessary system objects when you create the queue manager, by including the CSQ4INSX, CSQ4INSR and CSQ4INSG samples in the CSQINP2 initialization input data set. For more information, see Task 13: Customize the initialization input data sets.

The attributes of the publish/subscribe system queues are shown in Table 7 on page 105.

Table 7. Attributes of publish/subscribe system queues	
Attribute	**Default value**
DEFPSIST	Yes
DEFSOPT	SHARED
MAXMSGL	Multi On Multiplatforms: The value of the MAXMSGL parameter of the ALTER QMGR command z/OS On z/OS: 4194304 (that is, 4 MB)
MAXDEPTH	999999999
SHARE	N/A
z/OS z/OS STGCLASS	This attribute is used only on z/OS platforms

Note: The only queue that contains messages put by applications is SYSTEM.INTER.QMGR.PUBS. **MAXDEPTH** is set to its maximum value for this queue to allow temporary build up of published messages during outages or times of excessive load. If the queue manager is running on a system where that depth of queue could not be contained, this should be adjusted.

Related tasks

Troubleshooting distributed publish/subscribe problems

Distributed publish/subscribe system queue errors

Errors can occur when distributed publish/subscribe queue manager queues are unavailable. This affects the propagation of subscription knowledge across the publish/subscribe network, and publication to subscriptions on remote queue managers.

If the fan-out request queue SYSTEM.INTER.QMGR.FANREQ is unavailable, the creation of a subscription might generate an error, and error messages will be written to the queue manager error log when proxy subscriptions need to be delivered to directly connected queue managers.

If the hierarchy relationship state queue SYSTEM.HIERARCHY.STATE is unavailable, an error message is written to the queue manager error log and the publish/subscribe engine is put into COMPAT mode. To view the publish/subscribe mode, use the command DISPLAY QMGR PSMODE.

If any other of the SYSTEM.INTER.QMGR queues are unavailable, an error message is written to the queue manager error log and, although function is not disabled, it is likely that publish/subscribe messages will build up on queues on this or remote queue managers.

If the publish/subscribe system queue or required transmission queue to a parent, child or publish/subscribe cluster queue manager is unavailable, the following outcomes occur:

- The publications are not delivered, and a publishing application might receive an error. For details of when the publishing application receives an error, see the following parameters of the **DEFINE TOPIC** command: **PMSGDLV**, **NPMSGDLV**, and **USEDLQ**.
- Received inter-queue manager publications are backed out to the input queue, and subsequently re-attempted. If the backout threshold is reached, the undelivered publications are placed on the dead letter queue. The queue manager error log will contain details of the problem.
- An undelivered proxy subscription is backed out to the fanout request queue, and subsequently attempted again. If the backout threshold is reached, the undelivered proxy subscription is not delivered to any connected queue manager, and is placed on the dead letter queue. The queue manager error log will contain details of the problem, including details of any necessary corrective administrative action required.

- Hierarchy relationship protocol messages fail, and the connection status is flagged as ERROR. To view the connection status, use the command **DISPLAY PUBSUB**.

Related tasks

Troubleshooting distributed publish/subscribe problems

Multi Planning your storage and performance requirements on Multiplatforms

You must set realistic and achievable storage, and performance goals for your IBM MQ system. Use the links to find out about factors that affect storage and performance on your platform.

The requirements vary depending on the systems that you are using IBM MQ on, and what components you want to use.

For the latest information about supported hardware and software environments, see System Requirements for IBM MQ.

IBM MQ stores queue manager data in the file system. Use the following links to find out about planning and configuring directory structures for use with IBM MQ:

- "Planning file system support on Multiplatforms" on page 109
- "Requirements for shared file systems on Multiplatforms" on page 109
- "Sharing IBM MQ files on Multiplatforms" on page 118
- Linux AIX "Directory structure on AIX and Linux systems" on page 120
- Windows "Directory structure on Windows systems" on page 129
- IBM i "Directory structure on IBM i" on page 132

Use the following links for information about system resources, shared memory, and process priority on AIX and Linux:

- Linux AIX "IBM MQ and UNIX System V IPC resources" on page 137
- AIX "Shared memory on AIX" on page 136
- Linux AIX "IBM MQ and UNIX Process Priority" on page 137

Use the following links for information about log files:

- "Choosing circular or linear logging on Multiplatforms" on page 135
- Calculating the size of the log

Related concepts

"Planning your IBM MQ environment on z/OS" on page 137
When planning your IBM MQ environment, you must consider the resource requirements for data sets, page sets, Db2, Coupling Facilities, and the need for logging, and backup facilities. Use this topic to plan the environment where IBM MQ runs.

Related tasks

"Planning an IBM MQ architecture" on page 5
When planning your IBM MQ environment, consider the support that IBM MQ provides for single and multiple queue manager architectures, and for point-to-point and publish/subscribe messaging styles. Also plan your resource requirements, and your use of logging and backup facilities.

Related reference

Hardware and software requirements on AIX and Linux

Hardware and software requirements on Windows

Multi Disk space requirements on Multiplatforms

The storage requirements for IBM MQ depend on which components you install, and how much working space you need.

Disk storage is required for the optional components you choose to install, including any prerequisite components they require. The total storage requirement also depends on the number of queues that you use, the number and size of the messages on the queues, and whether the messages are persistent. You also require archiving capacity on disk, tape, or other media, as well as space for your own application programs.

The following tables show the approximate disk space required when you install various combinations of the product on different platforms. (Values are rounded up to the nearest 5 MB, where a MB is 1,048,576 bytes.)

- LTS "Disk space requirements for Long Term Support" on page 107
- CD "Disk space requirements for Continuous Delivery" on page 108

Disk space requirements for Long Term Support

V 10.0.0 LTS

Table 8. IBM MQ disk space requirements for Multiplatforms for Long Term Support

Platform	Client installation "1" on page 107	Server installation "2" on page 107	Full installation "3" on page 107
AIX AIX	335 MB	375 MB	1810 MB
IBM i IBM i (see Additional notes for IBM i)	485 MB	845 MB	1965 MB
Linux Linux for x86-64	270 MB	295 MB	2010 MB
Linux Linux on Power® Systems - Little Endian	170 MB	190 MB	1400 MB
Linux Linux on IBM Z®	255 MB	290 MB	1485 MB
Windows Windows (64 bit installation) "4" on page 108	295 MB	425 MB	2310 MB

Notes:

1. A client installation includes the following components:
   - Runtime
   - Client
2. A server installation includes the following components:
   - Runtime
   - Server
3. A full installation includes all available components.

4. Windows Not all the components listed here are installable features on Windows systems; their functionality is sometimes included in other features. See IBM MQ features for Windows systems.

Additional notes for IBM i: IBM i

1. On IBM i you cannot separate the native client from the server. The server figure in the table is for 5724H72*BASE without Java, together with the English Language Load (2924). There are 22 possible unique language loads.
2. The figure in the table is for the native client 5725A49 *BASE without Java.
3. Java and JMS classes can be added to both server and client bindings. If you want to include these features add 110 MB.
4. Adding samples source to either the client or server adds an extra 10 MB.
5. Adding samples to Java and JMS classes adds an extra 5 MB.

Disk space requirements for Continuous Delivery

CD V 10.0.0

Table 9. IBM MQ disk space requirements for Multiplatforms for Continuous Delivery

Platform/CD release	Client installation "1" on page 108	Server installation "2" on page 108	Full installation "3" on page 108
AIX **AIX**			
V 10.0.0 IBM MQ 10.0.0	390 MB	450 MB	1815 MB
Linux **Linux for x86-64 (64 bit)**			
V 10.0.0 IBM MQ 10.0.0	300 MB	305 MB	2010 MB
Linux **Linux on Power Systems - Little Endian**			
V 10.0.0 IBM MQ 10.0.0	185 MB	210 MB	1400 MB
Linux **Linux for IBM Z**			
V 10.0.0 IBM MQ 10.0.0	270 MB	290 MB	1485 MB
Windows **Windows (64 bit installation)** "4" on page 108			
V 10.0.0 IBM MQ 10.0.0	315 MB	425 MB	2310 MB

Notes:

1. A client installation includes the following components:
   - Runtime
   - Client
2. A server installation includes the following components:
   - Runtime
   - Server
3. A full installation includes all available components.
4. Windows Not all the components listed here are installable features on Windows systems; their functionality is sometimes included in other features. See IBM MQ features for Windows systems.

Related concepts

IBM MQ components and features

Multi Planning file system support on Multiplatforms

Queue manager data is stored in the file system. A queue manager makes use of file system locking to prevent multiple instances of a multi-instance queue manager being active at the same time.

Shared file systems

Shared file systems enable multiple systems to access the same physical storage device concurrently. Corruption would occur if multiple systems accessed the same physical storage device directly without some means of enforcing locking and concurrency control. Operating systems provide local file systems with locking and concurrency control for local processes; network file systems provide locking and concurrency control for distributed systems.

Historically, networked file systems have not performed fast enough, or provided sufficient locking and concurrency control, to meet the requirements for logging messages. Today, networked file systems can provide good performance, and implementations of reliable network file system protocols such as *RFC 3530, Network File System (NFS) version 4 protocol*, meet the requirements for logging messages reliably.

Shared file systems and IBM MQ

Queue manager data for a multi-instance queue manager is stored in a shared network file system. On AIX, Linux, and Windows systems, the queue manager's data files and log files must be placed in shared network file system. IBM i On IBM i, journals are used instead of log files, and journals cannot be shared. Multi-instance queue managers on IBM i use journal replication, or switchable journals, to make journals available between different queue manager instances.

IBM MQ uses locking to prevent multiple instances of the same multi-instance queue manager being active at the same time. The same locking also ensures that two separate queue managers cannot inadvertently use the same set of queue manager data files. Only one instance of a queue manager can have its lock at a time. Consequently, IBM MQ does support queue manager data stored on networked storage accessed as a shared file system.

Because not all the locking protocols of network file systems are robust, and because a file system might be configured for performance rather than data integrity, you must run the **amqmfsck** command to test whether a network file system will control access to queue manager data and logs correctly. This command applies only to UNIX, Linux and IBM i systems. On Windows, there is only one supported network file system and the **amqmfsck** command is not required.

Related tasks

“Verifying shared file system behavior on Multiplatforms” on page 111
Run **amqmfsck** to check whether a shared file system on AIX, Linux , or IBM i meets the requirements for storing the queue manager data of a multi-instance queue manager. (The only requirement for a Windows configuration is that it uses SMB 3 for shared storage provision.)

Multi Requirements for shared file systems on Multiplatforms

Shared files systems must provide data write integrity, guaranteed exclusive access to files and release locks on failure to work reliably with IBM MQ.

Requirements that a shared file system must meet

There are three fundamental requirements that a shared file system must meet to work reliably with IBM MQ:

1. Data write integrity

   Data write integrity is sometimes called *Write through to disk on flush*. The queue manager must be able to synchronize with data being successfully committed to the physical device. In a transactional system, you need to be sure that some writes have been safely committed before continuing with other processing.

More specifically, IBM MQ for AIX or Linux platforms use the *O_SYNC* open option and the `fsync()` system call to explicitly force writes to recoverable media, and the write operation is dependent upon these options operating correctly.

Attention: Linux You should mount the file system with the `async` option, which still supports the option of synchronous writes and gives better performance than the `sync` option.

Note, however, that if the file system has been exported from Linux, you must still export the file system using the `sync` option.

2. Guaranteed exclusive access to files

   In order to synchronize multiple queue managers, there needs to be a mechanism for a queue manager to obtain an exclusive lock on a file.

3. Release locks on failure

   If a queue manager fails, or if there is a communication failure with the file system, files locked by the queue manager need to be unlocked and made available to other processes without waiting for the queue manager to be reconnected to the file system.

A shared file system must meet these requirements for IBM MQ to operate reliably. If it does not, the queue manager data and logs get corrupted when using the shared file system in a multi-instance queue manager configuration.

For multi-instance queue managers on Microsoft Windows, the networked storage must be accessed by the Server Message Block (SMB) protocol used by Microsoft Windows networks. The Server Message Block (SMB) client does not meet the IBM MQ requirements for locking semantics on platforms other than Microsoft Windows, so multi-instance queue managers running on platforms other than Microsoft Windows must not use Server Message Block (SMB) as their shared file system.

For multi-instance queue managers on other supported platforms, the storage must be accessed by a network file system protocol which is Posix-compliant and supports lease-based locking. Network File System 4 satisfies this requirement. Older file systems, such as Network File System Version 3, which do not have a reliable mechanism to release locks after a failure, must not be used with multi-instance queue managers.

Checks on whether the shared file system meets the requirements

You must check whether the shared file system you plan to use meets these requirements. You must also check whether the file system is correctly configured for reliability. Shared file systems sometimes provide configuration options to improve performance at the expense of reliability.

For further information, see Testing statement for IBM MQ multi-instance queue manager file systems.

Under normal circumstances IBM MQ operates correctly with attribute caching and it is not necessary to disable caching, for example by setting NOAC on an NFS mount. Attribute caching can cause issues when multiple file system clients are contending for write access to the same file on the file system server, as the cached attributes used by each client might not be the same as those attributes on the server. An example of files accessed in this way are queue manager error logs for a multi-instance queue manager. The queue manager error logs might be written to by both an active and a standby queue manager instance and cached file attributes might cause the error logs to grow larger than expected, before rollover of the files occurs.

To help to check the file system, run the task Verifying shared file system behavior. This task checks if your shared file system meets requirements 2 and 3. You need to verify requirement 1 in your shared file system documentation, or by experimenting with logging data to the disk.

Disk faults can cause errors when writing to disk, which IBM MQ reports as First Failure Data Capture errors. You can run the file system checker for your operating system to check the shared file system for any disk faults. For example:

- Linux AIX On AIX and Linux the file system checker is called fsck.

- Windows On Windows platforms the file system checker is called CHKDSK, or SCANDISK.

NFS server security

Notes:

- You cannot use the **nosuid** or **noexec** options for a mount point that is used to hold the IBM MQ installation directory. This is because IBM MQ includes setuid/setgid executable programs, and these must not be prevented from running properly.
- When you put queue manager data only on a Network File System (NFS) server, you can use the following three options with the mount command to make the system secure, with no harmful impact to the running of the queue manager:

  noexec
  By using this option, you stop binary files from being run on the NFS, which prevents a remote user from running unwanted code on the system.

  nosuid
  By using this option, you prevent the use of the set-user-identifier and set-group-identifier bits, which prevents a remote user from gaining higher privileges.

  nodev
  By using this option, you stop character and block special devices from being used or defined, which prevents a remote user from getting out of a chroot jail.

Linux IBM i AIX ***Verifying shared file system behavior on Multiplatforms***

Run **amqmfsck** to check whether a shared file system on AIX, Linux , or IBM i meets the requirements for storing the queue manager data of a multi-instance queue manager. (The only requirement for a Windows configuration is that it uses SMB 3 for shared storage provision.)

Before you begin

You need a server with networked storage, and two other servers connected to it that have IBM MQ installed. You must have administrator (root) authority to configure the file system, and be an IBM MQ Administrator to run **amqmfsck**.

About this task

"Requirements for shared file systems on Multiplatforms" on page 109 describes the file system requirements for using a shared file system with multi-instance queue managers. The IBM MQ technote Testing statement for IBM MQ multi-instance queue manager file systems lists the shared file systems that IBM has already tested with. The procedure in this task describes how to test a file system to help you assess whether an unlisted file system maintains data integrity.

Failover of a multi-instance queue manager can be triggered by hardware or software failures, including networking problems which prevent the queue manager writing to its data or log files. Mainly, you are interested in causing failures on the file server. But you must also cause the IBM MQ servers to fail, to test any locks are successfully released. To be confident in a shared file system, test all of the following failures, and any other failures that are specific to your environment:

1. Shutting down the operating system on the file server including syncing the disks.
2. Halting the operating system on the file server without syncing the disks.
3. Pressing the reset button on each of the servers.
4. Pulling the network cable out of each of the servers.
5. Pulling the power cable out of each of the servers.
6. Switching off each of the servers.

Create the directory on the networked storage that you are going to use to share queue manager data and logs. The directory owner must be an IBM MQ Administrator, or in other words, a member of the `mqm` group on AIX and Linux. The user who runs the tests must have IBM MQ Administrator authority.

Use the example of exporting and mounting a file system in Creating a multi-instance queue manager on Linux or Creating a multi-instance queue manager using journal mirroring and NetServer on IBM i to help you through configuring the file system. Different file systems require different configuration steps. Read the file system documentation.

Note: Run the IBM MQ MQI client sample program **amqsfhac** in parallel with **amqmfsck** to demonstrate that a queue manager maintains message integrity during a failure.

Procedure

In each of the checks, cause all the failures in the previous list while the file system checker is running. If you intend to run **amqsfhac** at the same time as **amqmfsck**, do the task, "Running amqsfhac to test message integrity" on page 116 in parallel with this task.

1. Mount the exported directory on the two IBM MQ servers.

   On the file system server create a shared directory `shared`, and a subdirectory to save the data for multi-instance queue managers, `qmdata`. For an example of setting up a shared directory for multi-instance queue managers on Linux, see Creating a multi-instance queue manager on Linux
2. Check basic file system behavior.

   On one IBM MQ server, run the file system checker with no parameters.

   On IBM MQ server 1:

   ```
   amqmfsck /shared/qmdata
   ```
3. Check concurrently writing to the same directory from both IBM MQ servers.

   On both IBM MQ servers, run the file system checker at the same time with the `-c` option.

   On IBM MQ server 1:

   ```
   amqmfsck -c /shared/qmdata
   ```

   On IBM MQ server 2:

   ```
   amqmfsck -c /shared/qmdata
   ```
4. Check waiting for and releasing locks on both IBM MQ servers.

   On both IBM MQ servers run the file system checker at the same time with the `-w` option.

   On IBM MQ server 1:

   ```
   amqmfsck -w /shared/qmdata
   ```

   On IBM MQ server 2:

   ```
   amqmfsck -w /shared/qmdata
   ```
5. Check for data integrity.

   a) Format the test file.

   Create a large file in the directory being tested. The file is formatted so that the subsequent phases can complete successfully. The file must be large enough that there is sufficient time to interrupt the second phase to simulate the failover. Try the default value of 262144 pages (1 GB). The program automatically reduces this default on slow file systems so that formatting completes in about 60 seconds

   On IBM MQ server 1:

   ```
   amqmfsck -f /shared/qmdata
   ```

The server responds with the following messages:

```
Formatting test file for data integrity test.

Test file formatted with 262144 pages of data.
```

b) Write data into the test file using the file system checker while causing a failure.

Run the test program on two servers at the same time. Start the test program on the server which is going to experience the failure, then start the test program on the server that is going to survive the failure. Cause the failure you are investigating.

The first test program stops with an error message. The second test program obtains the lock on the test file and writes data into the test file starting where the first test program left off. Let the second test program run to completion.

Table 10. Running the data integrity check on two servers at the same time

IBM MQ server 1	**IBM MQ server 2**
`amqmfsck -a /shared/qmdata`	

`Please start this program on a second machine`
`with the same parameters.`
 	
`File lock acquired.`
 	
`Start a second copy of this program with the same parameters on another server.`
 	
`Writing data into test file.`
 	
`To increase the effectiveness of the test, interrupt the writing by ending the process,`
`temporarily breaking the network connection`
`to the networked storage,`
`rebooting the server or turning off the power.`
	`amqmfsck -a /shared/qmdata`
	
	`Waiting for lock...`
	
	`Waiting for lock...`
	
	`Waiting for lock...`
	
	`Waiting for lock...`
	
	`Waiting for lock...`
	
	`Waiting for lock...`

Turn the power off here.	

	`File lock acquired.`
	
	`Reading test file`
	
	`Checking the integrity of the data read.`
	
	`Appending data into the test file after data already found.`
	
	`The test file is full of data.`
	`It is ready to be inspected for data integrity.`

The timing of the test depends on the behavior of the file system. For example, it typically takes 30 - 90 seconds for a file system to release the file locks obtained by the first program following a

power outage. If you have too little time to introduce the failure before the first test program has filled the file, use the -x option of **amqmfsck** to delete the test file. Try the test from the start with a larger test file.

c) Verify the integrity of the data in the test file.

On IBM MQ server 2:

```
amqmfsck -i /shared/qmdata
```

The server responds with the following messages:

```
File lock acquired

Reading test file checking the integrity of the data read.

The data read was consistent.

The tests on the directory completed successfully.
```

6. Delete the test files.

On IBM MQ server 2:

```
amqmfsck -x /shared/qmdata

Test files deleted.
```

The server responds with the message:

```
Test files deleted.
```

Results

The program returns an exit code of zero if the tests complete successfully, and non-zero otherwise.

Examples

The first set of three examples shows the command producing minimal output.

Successful test of basic file locking on one server

```
> amqmfsck /shared/qmdata
The tests on the directory completed successfully.
```

Failed test of basic file locking on one server

```
> amqmfsck /shared/qmdata
AMQ6245: Error Calling 'write()[2]' on file '/shared/qmdata/amqmfsck.lck' error '2'.
```

Successful test of locking on two servers

Table 11. Successful locking on two servers

IBM MQ server 1	IBM MQ server 2

`> amqmfsck -w /shared/qmdata`
`Please start this program on a second`
`machine with the same parameters.`
`Lock acquired.`
`Press Return`
`or terminate the program to release the lock.`	

Table 11. Successful locking on two servers (continued)

IBM MQ server 1	IBM MQ server 2
	`> amqmfsck -w /shared/qmdata` `Waiting for lock...`
`[ Return pressed ]` `Lock released.`	
	`Lock acquired.` `The tests on the directory completed successfully`

The second set of three examples shows the same commands using verbose mode.

Successful test of basic file locking on one server

```
> amqmfsck -v /shared/qmdata
System call: stat("/shared/qmdata")'
System call: fd = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fchmod(fd, 0666)
System call: fstat(fd)
System call: fcntl(fd, F_SETLK, F_WRLCK)
System call: write(fd)
System call: close(fd)
System call: fd = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fcntl(fd, F_SETLK, F_WRLCK)
System call: close(fd)
System call: fd1 = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fcntl(fd1, F_SETLK, F_RDLCK)
System call: fd2 = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fcntl(fd2, F_SETLK, F_RDLCK)
System call: close(fd2)
System call: write(fd1)
System call: close(fd1)
The tests on the directory completed successfully.
```

Failed test of basic file locking on one server

```
> amqmfsck -v /shared/qmdata
System call: stat("/shared/qmdata")
System call: fd = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fchmod(fd, 0666)
System call: fstat(fd)
System call: fcntl(fd, F_SETLK, F_WRLCK)
System call: write(fd)
System call: close(fd)
System call: fd = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fcntl(fd, F_SETLK, F_WRLCK)
System call: close(fd)
System call: fd = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fcntl(fd, F_SETLK, F_RDLCK)
System call: fdSameFile = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fcntl(fdSameFile, F_SETLK, F_RDLCK)
System call: close(fdSameFile)
System call: write(fd)
AMQxxxx: Error calling 'write()[2]' on file '/shared/qmdata/amqmfsck.lck', errno 2
(Permission denied).
```

Successful test of locking on two servers

Table 12. Successful locking on two servers - verbose mode

IBM MQ server 1	IBM MQ server 2

`> amqmfsck -wv /shared/qmdata`
`Calling 'stat("/shared/qmdata")'`
`Calling 'fd = open("/shared/qmdata/`
`amqmfsck.lkw",`
`O_EXCL | O_CREAT | O_RDWR, 0666)'`
`Calling 'fchmod(fd, 0666)'`
`Calling 'fstat(fd)'`
`Please start this program on a second`
`machine with the same parameters.`
`Calling 'fcntl(fd, F_SETLK, F_WRLCK)'`
`Lock acquired.`
`Press Return`
`or terminate the program to release the lock.`	

	`> amqmfsck -wv /shared/qmdata`
	`Calling 'stat("/shared/qmdata")'`
	`Calling 'fd = open("/shared/qmdata/`
	`amqmfsck.lkw",`
	`O_EXCL | O_CREAT | O_RDWR,0666)'`
	`Calling 'fd = open("/shared/qmdata/amqmfsck.lkw,`
	`O_RDWR, 0666)'`
	`Calling 'fcntl(fd, F_SETLK, F_WRLCK)`
	`'Waiting for lock...`

`[ Return pressed ]`
`Calling 'close(fd)'`
`Lock released.`	

	`Calling 'fcntl(fd, F_SETLK, F_WRLCK)'`
	`Lock acquired.`
	`The tests on the directory completed`
	`successfully`

Related reference

High availability sample programs

Multi

Running amqsfhac to test message integrity

Run the IBM MQ MQI client sample program **amqsfhac** in parallel with **amqmfsck** to demonstrate that a queue manager maintains message integrity during a failure.

Before you begin

You require four servers for this test. Two servers for the multi-instance queue manager, one for the file system, and one for running **amqsfhac** as an IBM MQ MQI client application.

Follow step “1” on page 112 in “Verifying shared file system behavior on Multiplatforms” on page 111 to set up the file system for a multi-instance queue manager.

About this task

The IBM MQ MQI client sample program **amqsfhac** checks that a queue manager using networked storage maintains data integrity following a failure. Run **amqsfhac** in parallel with **amqmfsck** to demonstrate that a queue manager maintains message integrity during a failure.

Procedure

1. Create a multi-instance queue manager on another server, QM1, using the file system you created in step “1” on page 112 in Procedure.

   See Create a multi-instance queue manager.
2. Start the queue manager on both servers making it highly available.

   On server 1:

   ```
   strmqm -x QM1
   ```

   On server 2:

   ```
   strmqm -x QM1
   ```
3. Set up the client connection to run **amqsfhac**.
   a) Use the procedure in *Verifying an IBM MQ installation* for the platform, or platforms, that your enterprise use to set up a client connection, or the example scripts in Reconnectable client samples.
   b) Modify the client channel to have two IP addresses, corresponding to the two servers running QM1.

      In the example script, modify:

      ```
      DEFINE CHANNEL(CHANNEL1) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
      CONNAME('LOCALHOST(2345)') QMNAME(QM1) REPLACE
      ```

      To:

      ```
      DEFINE CHANNEL(CHANNEL1) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
      CONNAME('server1(2345),server2(2345)') QMNAME(QM1) REPLACE
      ```

      where server1 and server2 are the host names of the two servers, and 2345 is the port that the channel listener is listening on. Usually this defaults to 1414. You can use 1414 with the default listener configuration.
4. Create two local queues on QM1 for the test.
   Run the following MQSC script:

   ```
   DEFINE QLOCAL(TARGETQ) REPLACE
   DEFINE QLOCAL(SIDEQ) REPLACE
   ```
5. Test the configuration with **amqsfhac**

   ```
   amqsfhac QM1 TARGETQ SIDEQ 2 2 2
   ```
6. Test message integrity while you are testing file system integrity.

   Run **amqsfhac** during step “5” on page 112 of “Verifying shared file system behavior on Multiplatforms” on page 111.

   ```
   amqsfhac QM1 TARGETQ SIDEQ 10 20 0
   ```

   If you stop the active queue manager instance, **amqsfhac** reconnects to the other queue manager instance once it has become active. Restart the stopped queue manager instance again, so that you can reverse the failure in your next test. You will probably need to increase the number of iterations based on experimentation with your environment so that the test program runs for sufficient time for the failover to occur.

Results

An example of running **amqsfhac** in step “6” on page 117 is shown in the following example. In this example, the test is a success.

```
Sample AMQSFHAC start
qmname = QM1
qname = TARGETQ
sidename = SIDEQ
transize = 10
iterations = 20
verbose = 0
Iteration 0
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
Resolving MQRC_CALL_INTERRUPTED
MQGET browse side tranid=14 pSideinfo->tranid=14
Resolving to committed
Iteration 7
Iteration 8
Iteration 9
Iteration 10
Iteration 11
Iteration 12
Iteration 13
Iteration 14
Iteration 15
Iteration 16
Iteration 17
Iteration 18
Iteration 19
Sample AMQSFHAC end
```

If the test detected a problem, the output would report the failure. In some test runs `MQRC_CALL_INTERRUPTED` might report "`Resolving to backed out`". It makes no difference to the result. The outcome depends on whether the write to disk was committed by the networked file storage before or after the failure took place.

Related reference

amqmfsck (file system check)

High availability sample programs

Multi Sharing IBM MQ files on Multiplatforms

Some IBM MQ files are accessed exclusively by an active queue manager, other files are shared.

IBM MQ files are split into program files and data files. Program files are typically installed locally on each server running IBM MQ. Queue managers share access to data files and directories in the default data directory. They require exclusive access to their own queue manager directory trees contained in each of the `qmgrs` and `log` directories shown in Figure 32 on page 119.

Figure 32 on page 119 is a high-level view of the IBM MQ directory structure. It shows the directories which can be shared between queue managers and made remote. The details vary by platform. The dotted lines indicate configurable paths.

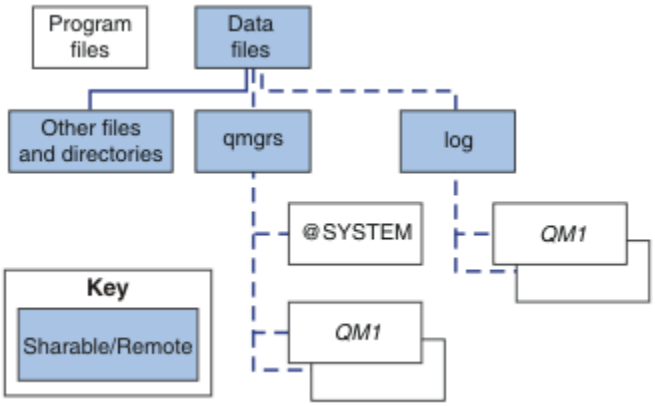


Figure 32. Overall view of IBM MQ directory structure

Program files

The program files directory is typically left in the default location, is local, and shared by all the queue managers on the server.

Data files

The data files directory is typically local in the default location, `/var/mqm` on AIX and Linux systems and configurable on installation on Windows. It is shared between queue managers. You can make the default location remote, but do not share it between different installations of IBM MQ. The `DefaultPrefix` attribute in the IBM MQ configuration points to this path.

`qmgrs`

There are two alternative ways to specify the location of queue manager data.

Using the `Prefix` attribute

The **`Prefix`** attribute specifies the location of the `qmgrs` directory. IBM MQ constructs the queue manager directory name from the queue manager name and creates it as a subdirectory of the `qmgrs` directory.

The **`Prefix`** attribute is located in the QueueManager stanza of the `mqs.ini` file, and is inherited from the value in the **`DefaultPrefix`** attribute of the All Queue Managers stanza. By default, for administrative simplicity, queue managers typically share the same `qmgrs` directory.

If you change the location of the `qmgrs` directory for any queue manager, you must change the value of its **`Prefix`** attribute.

The **`Prefix`** attribute for the `QM1` directory in Figure 32 on page 119 for a AIX and Linux platform is this:

```
Prefix=/var/mqm
```

Using the `DataPath` attribute

The **`DataPath`** attribute specifies the location of the queue manager data directory.

The **`DataPath`** attribute specifies the complete path, including the name of the queue manager data directory. The **`DataPath`** attribute is unlike the **`Prefix`** attribute, which specifies an incomplete path to the queue manager data directory.

The **`DataPath`** attribute, if it is specified, is located in the QueueManager stanza stanza of the `mqs.ini` file. If it has been specified, it takes precedence over any value in the **`Prefix`** attribute.

If you change the location of the queue manager data directory for any queue manager you must change the value of the `DataPath` attribute.

The DataPath attribute for the QM1 directory in Figure 32 on page 119, for a Linux or AIX platform, is as follows:

```
DataPath=/var/mqm/qmgrs/QM1
```

log

The log directory is specified separately for each queue manager in the Log stanza in the queue manager configuration. The queue manager configuration is in qm.ini.

***DataPath*/*QmgrName*/@IPCC subdirectories**

The *DataPath*/*QmgrName*/@IPCC subdirectories are in the shared directory path. They are used to construct the directory path for IPC file system objects. They need to distinguish the namespace of a queue manager when a queue manager is shared between systems.

The IPC file system objects have to be distinguished by system. A subdirectory, for each system the queue manager runs on, is added to the directory path, see Figure 33 on page 120.

```
DataPath/QmgrName/@IPCC/esem/myHostName/
```

Figure 33. Example IPC subdirectory

myHostName is up to the first 20 characters of the host name returned by the operating system. On some systems, the host name might be up to 64 characters in length before truncation. The generated value of *myHostName* might cause a problem for two reasons:

1. The first 20 characters are not unique.
2. The host name is generated by a DHCP algorithm that does not always allocate the same host name to a system.

In these cases, set *myHostName* using the environment variable **MQS_IPC_HOST**; see Figure 34 on page 120.

```
export MQS_IPC_HOST= myHostName
```

*Figure 34. Example: setting **MQS_IPC_HOST***

Other files and directories

Other files and directories, such as the directory containing trace files, and the common error log, are normally shared and kept on the local file system.

With support of shared file systems, IBM MQ manages exclusive access to these files using file system locks. A file system lock allows only one instance of a particular queue manager to be active at a time.

When you start the first instance of a particular queue manager it takes ownership of its queue manager directory. If you start a second instance, it can only take ownership if the first instance has stopped. If the first queue manager is still running, the second instance fails to start, and reports the queue manager is running elsewhere. If the first queue manager has stopped, then the second queue manager takes over ownership of the queue manager files and becomes the running queue manager.

You can automate the procedure of the second queue manager taking over from the first. Start the first queue manager with the strmqm -x option that permits another queue manager to take over from it. The second queue manager then waits until the queue manager files are unlocked before attempting to take over ownership of the queue manager files, and start.

Linux AIX Directory structure on AIX and Linux systems

The IBM MQ directory structure on AIX and Linux systems can be mapped to different file systems for easier management, better performance, and better reliability.

Use the flexible directory structure of IBM MQ to take advantage of shared file systems for running multi-instance queue managers.

Use the command **crtmqm** *QM1* to create the directory structure shown in Figure 35 on page 121 where R is the release of the product. It is a typical directory structure for a queue manager created on an IBM MQ system. Some directories, files and .ini attribute settings are omitted for clarity, and another queue manager name might be altered by mangling. The names of the file systems vary on different systems.

In a typical installation, every queue manager that you create points to common `log` and `qmgrs` directories on the local file system. In a multi-instance configuration, the `log` and `qmgrs` directories are on a network file system shared with another installation of IBM MQ.

V 10.0.0 A Native HA CRR configuration also has a backup log directory that is used during rebase operations. When a rebase happens, the existing log directory is renamed as *QMname*`-backup` and the log directory is recreated using the same name and permissions to receive the rebase log data. If a custom log directory was specified when the queue manager was created (or subsequently) then the backup log directory takes the name *customlogname*`-backup`. The backup directory is deleted after the rebase has succeeded and it is no longer required, you should not delete it manually.

Figure 35 on page 121 shows the default configuration for IBM MQ v7.R on AIX where R is the release of the product. For examples of alternative multi-instance configurations, see"Example directory configurations on AIX and Linux systems" on page 125.

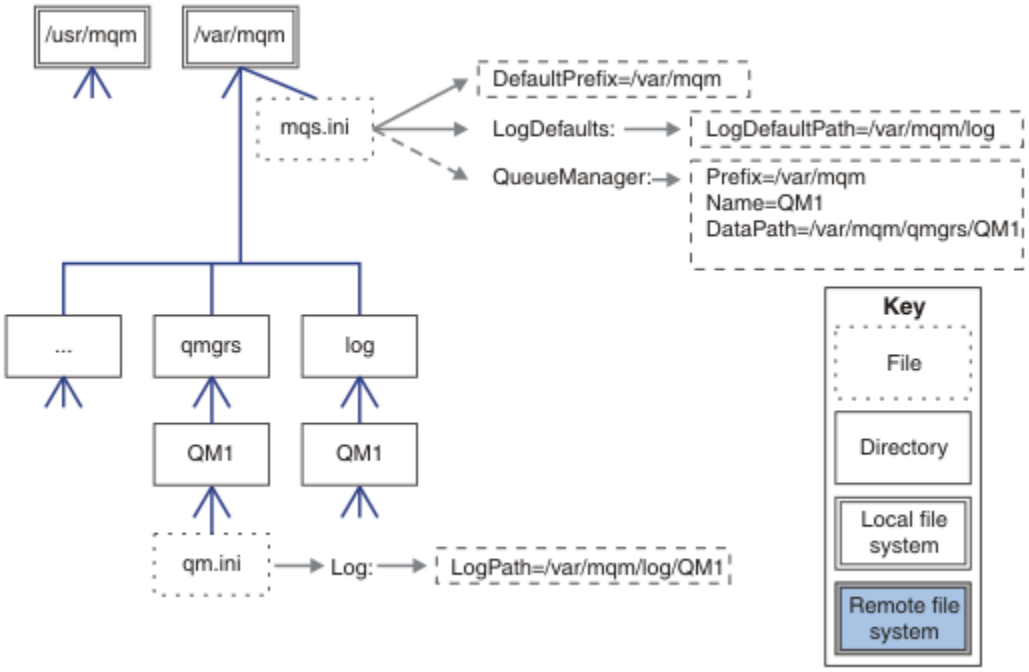


Figure 35. Example default IBM MQ directory structure for AIX and Linux systems

The product executable files are installed into `/usr/mqm` on AIX and `/opt/mqm` on the other systems, by default. The working directories are created under the `/var/mqm` directory.

Note: If you created the `/var/mqm` file system prior to installing IBM MQ , ensure that the mqm user has full directory permissions, for example, file mode 755.

Note: The `/var/mqm/errors` directory should be a separate filesystem to prevent FFDCs produced by the queue manager from filling the filesystem that contains `/var/mqm`.

See Creating file systems on AIX and Linux systems for more information.

The `log` and `qmgrs` directories are shown in their default locations as defined by the default values of the `LogDefaultPath` and `DefaultPrefix` attributes in the `mqs.ini` file. When a queue manager is created, by default the queue manager data directory is created in *DefaultPrefix*`/qmgrs`, and the log file directory in *LogDefaultPath*`/log`. `LogDefaultPath` and `DefaultPrefix` only effects where queue managers and log files are created by default. The actual location of a queue manager directory is saved in the `mqs.ini` file, and the location of the log file directory is saved in the `qm.ini` file.

The log file directory for a queue manager is defined in the `qm.ini` file in the `LogPath` attribute. Use the `-ld` option on the **crtmqm** command to set the `LogPath` attribute for a queue manager; for example, **crtmqm** `-ld` *LogPath* `QM1` . If you omit the `ld` parameter the value of `LogDefaultPath` is used instead.

The queue manager data directory is defined in the `DataPath` attribute in the `QueueManager` stanza in the `mqs.ini` file. Use the `-md` option on the **crtmqm** command to set the `DataPath` for a queue manager; for example, **crtmqm** `-` `md` *DataPath* `QM1` . If you omit the `md` parameter the value of the `DefaultPrefix` or `Prefix` attribute is used instead. `Prefix` takes precedence over `DefaultPrefix`.

Typically, create `QM1` specifying both the log and data directories in a single command.

```
 crtmqm
-md DataPath -ld
LogPath QM1
```

You can modify the location of a queue manager log and data directories of an existing queue manager by editing the `DataPath` and `LogPath` attributes in the `qm.ini` file when the queue manager is stopped.

The path to the `errors` directory, like the paths to all the other directories in `/var/mqm`, is not modifiable. However the directories can be mounted on different file systems, or symbolically linked to different directories.

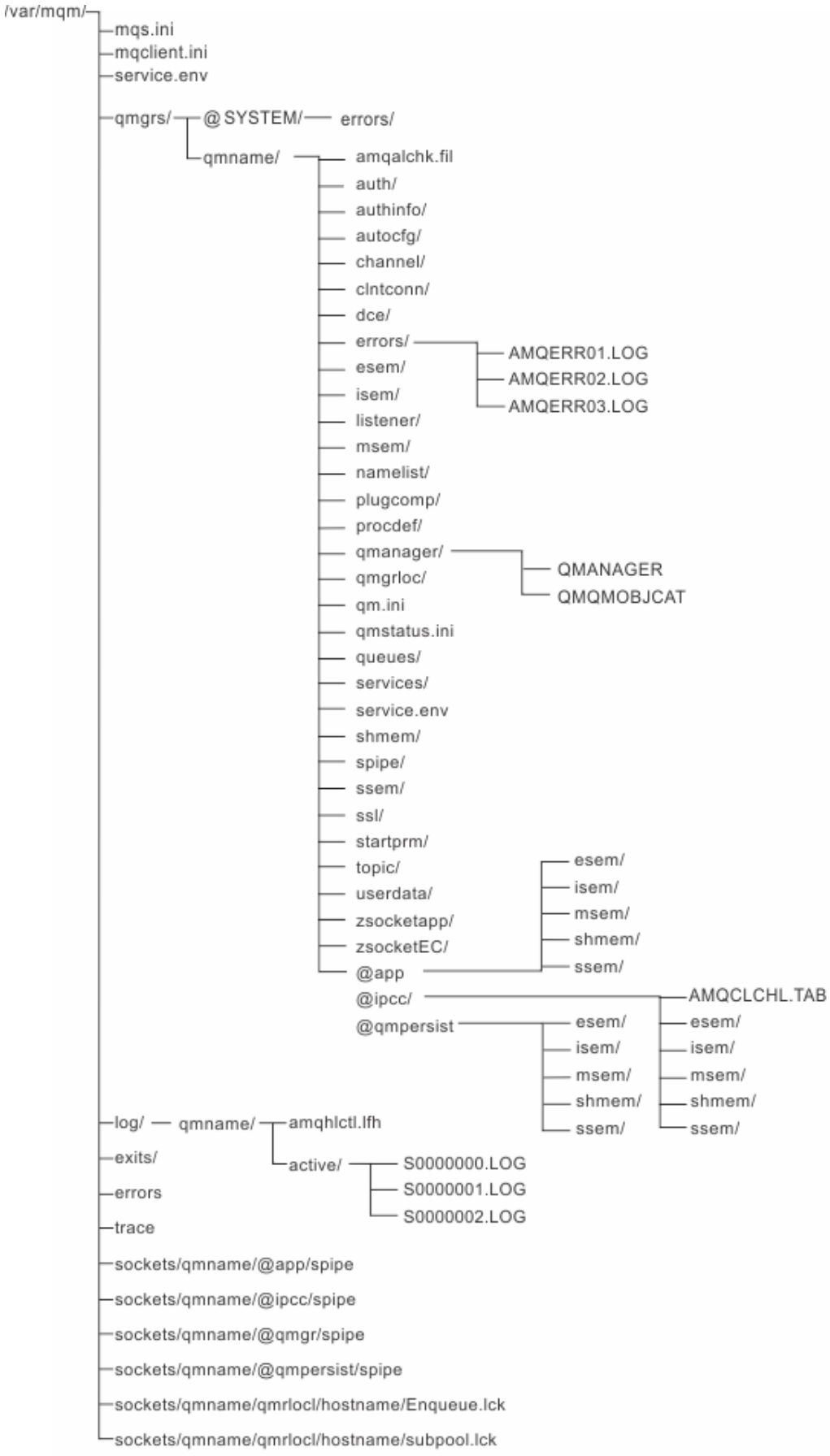
Linux AIX *Directory content on AIX and Linux systems*

Content of the directories associated with a queue manager.

For information about the location of the product files, see Choosing an installation location

For information about alternative directory configurations, see “Planning file system support on Multiplatforms” on page 109.

The following directory structure is representative of IBM MQ after a queue manager has been in use for some time. The actual structure that you have depends on which operations have occurred on the queue manager.


/var/mqm/
mqs.ini
mqclient.ini
service.env
qmgrs/
@SYSTEM/
errors/
qmname/
amqalchk.fil
auth/
authinfo/
autocfg/
channel/
clntconn/
dce/
errors/
AMQERR01.LOG
AMQERR02.LOG
AMQERR03.LOG
esem/
isem/
listener/
msem/
namelist/
plugcomp/
procdef/
qmanager/
QMANAGER
QMQMOBJCAT
qmgrloc/
qm.ini
qmstatus.ini
queues/
services/
service.env
shmem/
spipe/
ssem/
ssl/
startprm/
topic/
userdata/
zsocketapp/
zsocketEC/
@app
esem/
isem/
msem/
shmem/
ssem/
@ipcc/
AMQCLCHL.TAB
esem/
isem/
msem/
shmem/
ssem/
@qmpersist
esem/
isem/
msem/
shmem/
ssem/
log/
qmname/
amqhlctl.lfh
active/
S0000000.LOG
S0000001.LOG
S0000002.LOG
exits/
errors
trace
sockets/qmname/@app/spipe
sockets/qmname/@ipcc/spipe
sockets/qmname/@qmgr/spipe
sockets/qmname/@qmpersist/spipe
sockets/qmname/qmrlocl/hostname/Enqueue.lck
sockets/qmname/qmrlocl/hostname/subpool.lck

/var/mqm/

The /var/mqm directory contains configuration files and output directories that apply to an IBM MQ installation as a whole, and not to an individual queue manager.

Table 13. Documented content of the /var/mqm directory on AIX and Linux

Directory or file name	Contents
mqs.ini	IBM MQ installation wide configuration file, read when a queue manager starts. File path modifiable using the **AMQ_MQS_INI_LOCATION** environment variable. Ensure this is set and exported in the shell in which the **strmqm** command is run.
mqclient.ini	Default client configuration file read by IBM MQ MQI client programs. File path modifiable using the **MQCLNTCF** environment variable.
service.env	Contains machine scope environment variables for a service process. File path fixed.
errors/	Machine scope error logs, and FFST files. Directory path fixed. See also, FFST: IBM MQ for UNIX and Linux systems.
sockets/	Contains information for each queue manager for system use only.
trace/	Trace files. Directory path fixed.
web/	mqweb server directory.
exits/	Default directory containing user channel exit programs. Location modifiable in ApiExit stanzas in the mqs.ini file.
exits64/	

/var/mqm/qmgrs/*qmname*/

/var/mqm/qmgrs/*qmname*/ contains directories and files for a queue manager. The directory is locked for exclusive access by the active queue manager instance. The directory path is directly modifiable in the mqs.ini file, or by using the **md** option of the **crtmqm** command.

Table 14. Documented contents of the /var/mqm/qmgrs/qmname directory on AIX and Linux

Directory or file name	Contents
qm.ini	Queue manager configuration file, read when a queue manager starts.
errors/	Queue manager scope error logs. *qmname* = @system contains channel-related messages for an unknown or unavailable queue manager.
@ipcc/AMQCLCHL.TAB	Default client channel control table, created by the IBM MQ server, and read by IBM MQ MQI client programs. File path modifiable using the **MQCHLLIB** and **MQCHLTAB** environment variables.

Table 14. Documented contents of the /var/mqm/qmgrs/qmname directory on AIX and Linux (continued)	
Directory or file name	**Contents**
qmanager	Queue manager object file: QMANAGER Queue manager object catalog: QMQMOBJCAT
authinfo/	Each object defined within the queue manager is associated with a file in these directories. The file name approximately matches the definition name; see, Understanding IBM MQ file names.
channel/	
clntconn/	
listener/	
namelist/	
procdef/	
queues/	
services/	
topics/	
...	Other directories used by IBM MQ, such as @ipcc, to be modified only by IBM MQ.
userdata/	Can be used to store the persistent state of applications (can be used by RDQM when moving queue managers to different nodes - see Storing persistent application status.)
***DataPath*\autocfg**	Used for automatic configuration

/var/mqm/log/*qmname*/

/var/mqm/log/*qmname*/ contains the queue manager log files. The directory is locked for exclusive access by the active queue manager instance. The directory path is modifiable in the qm.ini file, or by using the **ld** option of the **crtmqm** command.

Table 15. Documented contents of the /var/mqm/log/qmname directory on AIX and Linux	
Directory or file name	**Contents**
amqhlctl.lfh	Log control file.
active/	This directory contains the log files numbered S0000000.LOG, S0000001.LOG, S0000002.LOG, and so on.

/opt/mqm

/opt/mqm is, by default, the installation directory on most platforms. See "Disk space requirements on Multiplatforms" on page 107 for more information on the amount of space that you need for the installation directory on the platform, or platforms, that your enterprise uses.

Linux AIX Example directory configurations on AIX and Linux systems

Examples of alternative file system configurations on AIX and Linux systems.

You can customize the IBM MQ directory structure in various ways to achieve a number of different objectives.

- Place the `qmgrs` and `log` directories on remote shared file systems to configure a multi-instance queue manager.
- Use separate file systems for the data and log directories, and allocate the directories to different disks, to improve performance by reducing I/O contention.
- Use faster storage devices for directories that have a greater effect on performance. Physical device latency is frequently a more important factor in the performance of persistent messaging than whether a device is mounted locally or remotely. The following list shows which directories are most and least performance sensitive.
  1. `log`
  2. `qmgrs`
  3. Other directories, including `/usr/mqm`
- Create the `qmgrs` and `log` directories on file systems that are allocated to storage with good resilience, such as a redundant disk array, for example.
- It is better to store the common error logs in `var/mqm/errors`, locally, rather than on a network file system, so that error relating to the network file system can be logged.

Figure 36 on page 126 is a template from which alternative IBM MQ directory structures are derived. In the template, dotted lines represent paths that are configurable. In the examples, the dotted lines are replaced by solid lines that correspond to the configuration information stored in the AMQ_MQS_INI_LOCATION environment variable, and in the `mqs.ini` and `qm.ini` files.

Note: The path information is shown as it appears in the `mqs.ini` or `qm.ini` files. If you supply path parameters in the **crtmqm** command, omit the name of the queue manager directory: the queue manager name is added to the path by IBM MQ.

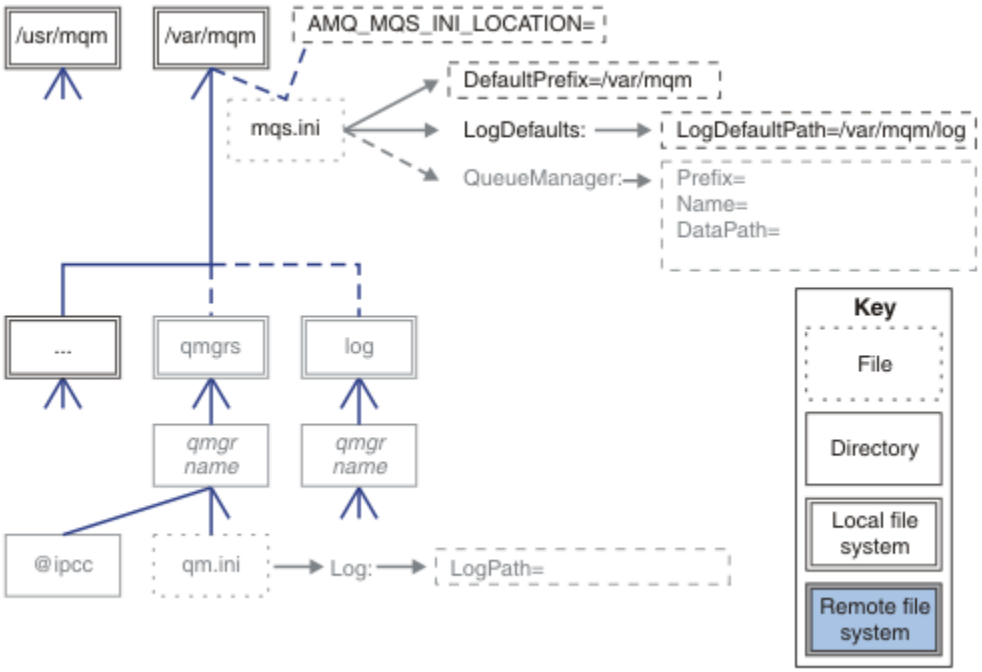


Figure 36. Directory structure pattern template

Typical directory structure for IBM MQ

Figure 37 on page 127 is the default directory structure created in IBM MQ by issuing the command **crtmqm** *QM1*.

The `mqs.ini` file has a stanza for the QM1 queue manager, created by referring to the value of `DefaultPrefix`. The `Log` stanza in the `qm.ini` file has a value for `LogPath`, set by reference to `LogDefaultPath` in `mqs.ini`.

Use the optional **crtmqm** parameters to override the default values of DataPath and LogPath.

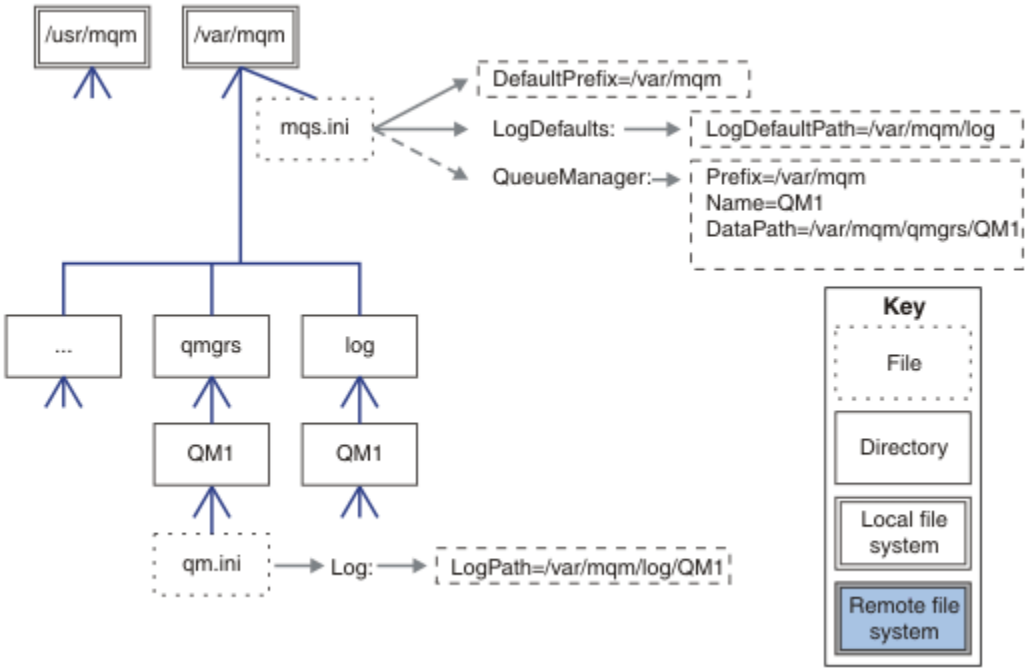


Figure 37. Example default IBM MQ directory structure for AIX and Linux systems

Share default qmgrs and log directories

An alternative to "Share everything " on page 128, is to share the qmgrs and log directories separately (Figure 38 on page 127). In this configuration, there is no need to set AMQ_MQS_INI_LOCATION as the default mqs.ini is stored in the local /var/mqm file system. The files and directories, such as mqclient.ini and mqserver.ini are also not shared.

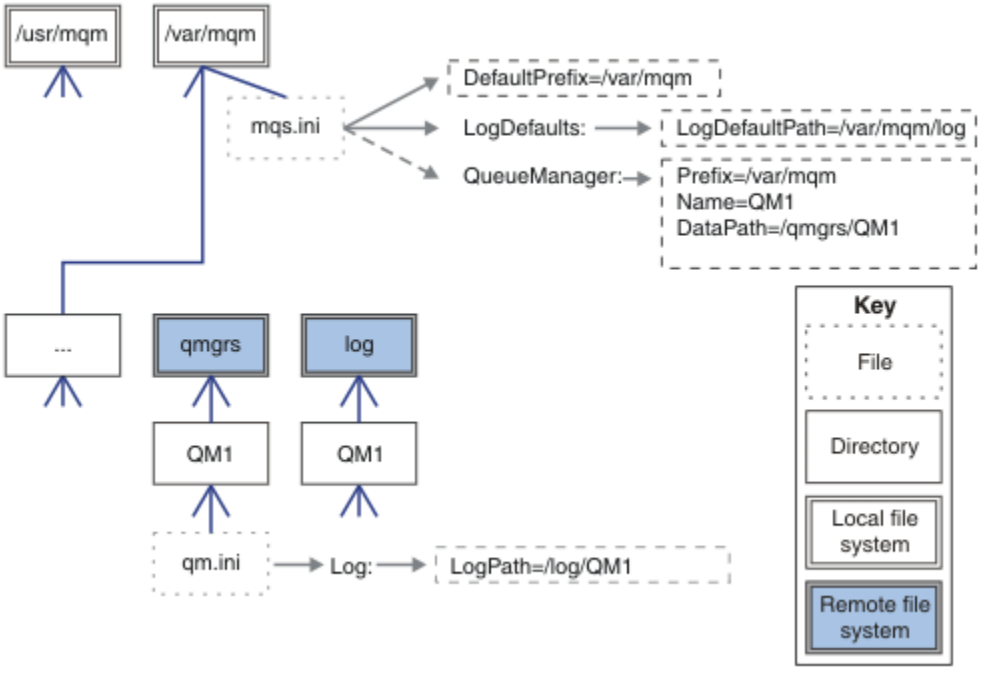


Figure 38. Share qmgrs and log directories

Share named qmgrs and log directories

The configuration in Figure 39 on page 128 places the log and qmgrs in a common named remote shared file system called /ha. The same physical configuration can be created in two different ways.

1. Set LogDefaultPath=/ha and then run the command, **crtmqm** - md */ha/qmgrs* QM1. The result is exactly as illustrated in Figure 39 on page 128.
2. Leave the default paths unchanged and then run the command, **crtmqm** - ld */ha/log* - md */ha/qmgrs* QM1.

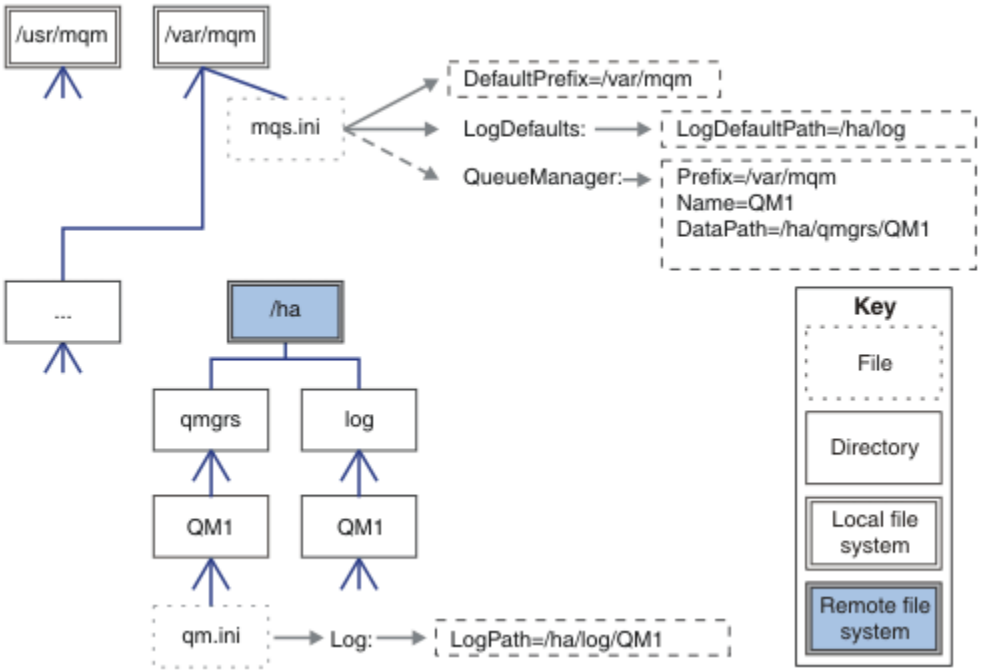


Figure 39. Share named qmgrs and log directories

Share everything

Figure 40 on page 129 is a simple configuration for system with fast networked file storage.

Mount /var/mqm as a remote shared file system. By default, when you start QM1, it looks for /var/mqm, finds it on the shared file system, and reads the mqs.ini file in /var/mqm. Rather than use the single /var/mqm/mqs.ini file for queue managers on all your servers, you can set the AMQ_MQS_INI_LOCATION environment variable on each server to point to different mqs.ini files.

Note: The contents of the generic error file in /var/mqm/errors/ are shared between queue managers on different servers.

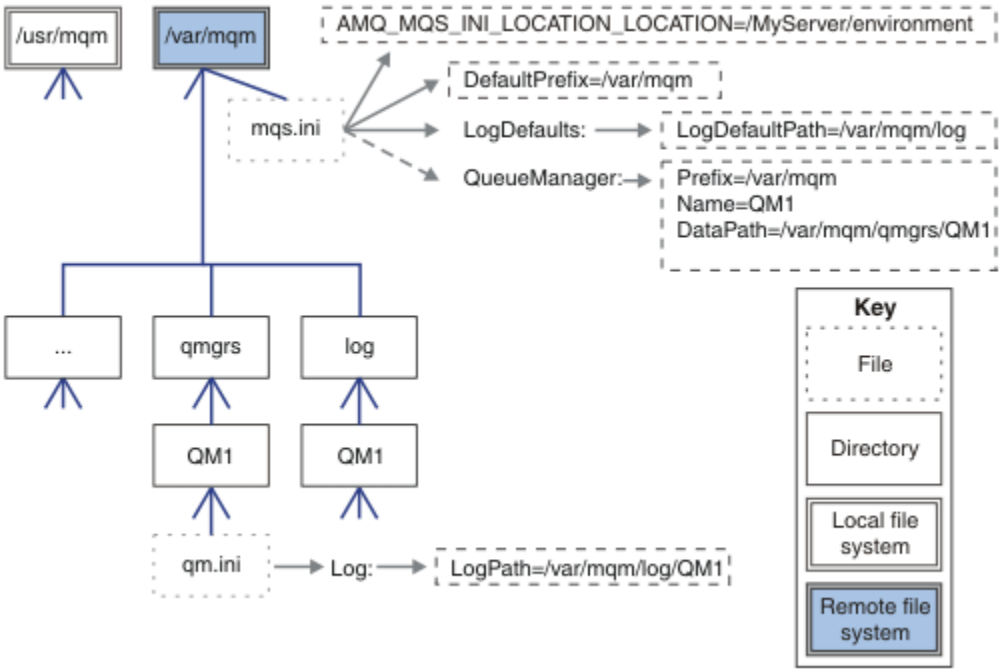


Figure 40. Share everything

Note that you cannot use this for multi-instance queue managers. The reason is that it is necessary for each host in a multi-instance queue manager to have its own local copy of `/var/mqm` to keep track of local data, such as semaphores and shared memory. These entities cannot be shared across hosts.

Windows Directory structure on Windows systems

How to find queue manager configuration information and directories on Windows.

The default directories for IBM MQ for Windows installation are:

Program directory
`C:\Program Files\IBM\MQ`

Data directory
`C:\ProgramData\IBM\MQ`

Important: Windows For Windows installations, the directories are as stated, unless there is a previous installation of the product that still contains registry entries or queue managers, or both. In this situation, the new installation uses the old data directory location. For more information, see Program and data directory locations.

If you want to know which installation directory and which data directory is being used, run the dspmqver command.

The installation directory is listed in the **InstPath** field and the data directory is listed in the **DataPath** field.

Running the **dspmqver** command displays, for example, the following information:

```
>dspmqver
Name:        IBM MQ
Version:     9.0.0.0
Level:       p900-L160512.4
BuildType:   IKAP - (Production)
Platform:    IBM MQ for Windows (x64 platform)
Mode:        64-bit
O/S:         Windows 7 Professional x64 Edition, Build 7601: SP1
InstName:    Installation1
InstDesc:
Primary:     Yes
```

```
InstPath:    C:\Program Files\IBM\MQ
DataPath:    C:\ProgramData\IBM\MQ
MaxCmdLevel: 900
LicenseType: Production
```

Multi-instance queue managers

To configure a multi-instance queue manager, the log and data directories must be placed on networked storage, preferably on a different server to any of the servers that are running instances of the queue manager.

Two parameters are provided on the **crtmqm** command, **-md** and **-ld**, to make it easier specify the location of the queue manager data and log directories. The effect of specifying the **-md** parameter is fourfold:

1. The `mqs.ini` stanza `QueueManager\`*`QmgrName`* contains a new variable, *DataPath*, which points to the queue manager data directory. Unlike the *Prefix* variable, the path includes the name of the queue manager directory.
2. The queue manager configuration information stored in the mqs.ini file is reduced to *Name*, *Prefix*, *Directory* and *DataPath*.

Windows *Directory content*

Lists the location and content of IBM MQ directories.

An IBM MQ configuration has three main sets of files and directories:

1. Executable, and other read-only files that are only updated when maintenance is applied. For example:
   - The readme file
   - The IBM MQ Explorer plug-in and help files
   - License files

   These files are described in Table 16 on page 130.
2. Potentially modifiable files and directories that are not specific to a particular queue manager. These files and directories are described in Table 17 on page 131.
3. Files and directories that are specific to each queue manager on a server. These files and directories are described in Table 18 on page 132.

Resource directories and files

The resource directories and files contain all the executable code and resources to run a queue manager. The variable, *FilePath*, in the installation specific IBM MQ configuration registry key, contains the path to the resource directories.

Table 16. Directories and files in the `FilePath` directory

File path	Contents
FilePath\bin	Commands and DLLs
FilePath\bin64	Commands and DLLs (64 bit)
FilePath\conv	Data conversion tables
FilePath\doc	Wizard help files
FilePath\MQExplorer	Explorer and Explorer help Eclipse plug-ins
FilePath\gskit8	Global security kit
FilePath\java	Java resources, including JRE
FilePath\licenses	License information

Table 16. Directories and files in the FilePath directory (continued)	
File path	**Contents**
FilePath\Non_IBM_License	License information
FilePath\properties	Used internally
FilePath\Tivoli	
FilePath\tools	Development resources and samples
FilePath\web	Described in IBM MQ Console and REST API installation component file structure for non-editable files.
FilePath\Uninst	Used internally
FilePath\README.TXT	Readme file

Directories not specific to a queue manager

Some directories contain files, such as trace files and error logs, that are not specific to a specific queue manager. The *DefaultPrefix* variable contains the path to these directories. *DefaultPrefix* is part of the AllQueueManagers stanza.

Table 17. Directories and files in DefaultPrefix directory	
File path	**Contents**
DefaultPrefix\config	Used internally
DefaultPrefix\conv	ccsid_part2.tbl and ccsid.tbl data conversion control file, described in Data conversion
DefaultPrefix\errors	Non queue manager error logs, AMQERR *nn*.LOG
DefaultPrefix\exits	Channel exit programs
DefaultPrefix\exits64	Channel exit programs (64 bit)
DefaultPrefix\ipc	Not used
DefaultPrefix\qmgrs	Described in Table 18 on page 132
DefaultPrefix\trace	Trace files
DefaultPrefix\web	Described in IBM MQ Console and REST API installation component file structure for user editable files
DefaultPrefix\amqmjpse.txt	Used internally

Queue manager directories

When you create a queue manager, a new set of directories, specific to the queue manager, is created.

If you create a queue manager with the **-md** *filepath* parameter, the path is stored in the *DataPath* variable in the queue manager stanza of the mqs.ini file. If you create a queue manager without setting the **-md** *filepath* parameter, the queue manager directories are created in the path stored in *DefaultPrefix*, and the path is copied into the *Prefix* variable in the queue manager stanza of the mqs.ini file.

Table 18. Directories and files in DataPath and Prefix\qmgrs\QmgrName directories	
File path	**Contents**
DataPath\@ipcc	Default location for AMQCLCHL.TAB, the client connection table.
DataPath\authinfo	Used internally.
DataPath\channel	
DataPath\clntconn	
DataPath\errors	Error logs, AMQERR *nn*.LOG
DataPath\listener	Used internally.
DataPath\namelist	
DataPath\plugcomp	
DataPath\procdef	
DataPath\qmanager	
DataPath\queues	
DataPath\services	
DataPath\ssl	
DataPath\startprm	
DataPath\topic	
DataPath\active	
DataPath\active.dat	
DataPath\amqalchk.fil	
DataPath\master	
DataPath\master.dat	
DataPath\qm.ini	Queue manager configuration
DataPath\qmstatus.ini	Queue manager status
DataPath\userdata	Can be used to store the persistent state of applications.
Prefix\qmgrs*QmgrName*	Used internally
Prefix\qmgrs\@SYSTEM	Not used
Prefix\qmgrs\@SYSTEM\errors	
*DataPath**autocfg*	Used for automatic configuration

IBM i Directory structure on IBM i

A description of the IFS is given, and the IBM MQ IFS directory structure is described for server, client, and Java.

The integrated file system (IFS) is a part of IBM i that supports stream input/output and storage management similar to personal computer, AIX and Linux operating systems, while providing an integrating structure over all information stored in the server.

On IBM i directory names begin with the character & (ampersand) instead of the character @ (at). For example, @system on IBM i is &system.

IFS root file system for IBM MQ server

When you install IBM MQ Server for IBM i, the following directories are created in the IFS root file system.

ProdData:

Overview

```
QIBM
   '-- ProdData
      '-- mqm
      '-- doc
      '-- inc
      '-- lib
      '-- samp
      '-- licenses
      '-- LicenseDoc
      '-- 5724H72_V8R0M0
```

/QIBM/ProdData/mqm
Subdirectories below this contain all the product data, for example, C++ classes, trace format files, and license files. Data in this directory is deleted and replaced each time the product is installed.

/QIBM/ProdData/mqm/doc
A Command Reference for the CL commands is provided in HTML format and installed here.

/QIBM/ProdData/mqm/inc
The header files for compiling your C or C++ programs.

/QIBM/ProdData/mqm/lib
Auxiliary files used by MQ.

/QIBM/ProdData/mqm/samp
Further samples.

/QIBM/ProdData/mqm/licenses
License files. The two files for each language are named like LA_ *xx* and LI_ *xx* where *xx* is the 2 character language identifier for each language supplied.

Also the following directory stores license agreements files:

/QIBM/ProdData/LicenseDoc/5724H72_V8R0M0
License files. The files are named like 5724H72_V8R0M0_ *xx* where *xx* is the 2 or 5 character language identifier for each language supplied.

UserData:

Overview

```
QIBM
   '-- UserData
      '-- mqm
      '-- errors
      '-- trace
      '-- qmgrs
      '-- &system
      '-- qmgrname1
      '-- qmgrname2
      '-- and so on
```

/QIBM/UserData/mqm
Subdirectories below this contain all user data relating to queue managers.

When you install the product, an mqs.ini file is created in directory /QIBM/UserData/mqm/ (unless it is already there from a previous installation).

When you create a queue manager, a qm.ini file is created in the directory /QIBM/UserData/mqm/qmgrs/ *QMGRNAME* / (where *QMGRNAME* is the name of the queue manager).

Data in the directories is retained when the product is deleted.

IFS root file system for IBM MQ MQI client

When you install IBM MQ MQI client for IBM i, the following directories created in the IFS root file system:

ProdData:

Overview

```
QIBM
   '-- ProdData
      '-- mqm
      '-- lib
```

/QIBM/ProdData/mqm
Subdirectories below this directory contain all the product data. Data in this directory is deleted and replaced each time the product is replaced.

UserData:

Overview

```
QIBM
   '-- UserData
      '-- mqm
      '-- errors
      '-- trace
```

/QIBM/UserData/mqm
Subdirectories below this directory contain all user data.

IFS root file system for IBM MQ Java

When you install IBM MQ Java on IBM i, the following directories are created in the IFS root file system:

ProdData:

Overview

```
QIBM
   '-- ProdData
      '-- mqm
      '-- java
      '--samples
      '-- bin
      '-- lib
```

/QIBM/ProdData/mqm/java
Subdirectories below this contain all the product data, including Java classes. Data in this directory is deleted and replaced each time the product is replaced.

/QIBM/ProdData/mqm/java/samples
Subdirectories below this contain all the sample Java classes and data.

Libraries created by server and client installations

Installation of the IBM MQ server or client creates the following libraries:

- QMQM

  The product library.

- QMQMSAMP

  The samples library (if you choose to install the samples).

- QMxxxx

  Server only.

  Each time that you create a queue manager, IBM MQ automatically creates an associated library, with a name like QMxxxx where xxxx is derived from the queue manager name. This library contains objects specific to the queue manager, including journals and associated receivers. By default the name of this library is derived from the name of the queue manager prefixed with the characters QM. For example, for a queue manager called TEST, the library would be called QMTEST.

  Note: When you create a queue manager, you can specify the name of its library if you want to. For example:

  ```
  CRTMQM MQMNAME(TEST) MQMLIB(TESTLIB)
  ```

  You can use the WRKLIB command to list all the libraries that IBM MQ for IBM i has created. Against the queue manager libraries, you will see the text QMGR: QMGRNAME. The format of the command is:

  ```
  WRKLIB LIB(QM*)
  ```

  These queue manager-associated libraries are retained when the product is deleted.

Multi Planning file system support for MFT on Multiplatforms

IBM MQ Managed File Transfer MFT agents can be used to transfer data to, and from, files on a file system. In addition to this, resource monitors running within an agent can be configured to monitor for files on a file system.

MFT has a requirement that these files are stored on a file system that supports locking. There are two reasons for this:

- An agent locks a file to ensure that it does not change once it has started reading data from it, or writing data to it.
- Resource monitors lock files to check that no other processes are currently using them.

Agents and resource monitors use the Java method **`FileChannel.tryLock()`** to perform locking, and the file system must be able to lock files when asked to do so using this call.

Important: The following file systems are not supported, as they do not meet MFT's technical requirements:

- GlusterFS
- NFS version 3

Multi Choosing circular or linear logging on Multiplatforms

In IBM MQ, you can choose circular or linear logging for your recovery logs. The following information gives you an overview of both types.

See Logging: Making sure that messages are not lost for more information about recovery logging. IBM MQ uses checkpointing to limit the volume of log replay required following a crash recovery. The key event that controls whether a log file is termed active or not is a `checkpoint`. For more information about transactions and checkpointing, see Using checkpointing to ensure complete recovery.

Advantages of circular logging

The main advantages of circular logging are that circular logging is:

- Easier to administer.

  Once you have configured circular logging correctly for your workload, no further administration is needed. Whereas, for linear logging, media images need to be recorded and log extents that are not required any more need to be archived or deleted.

- Better performing

  Circular logging performs better than linear logging, because circular logging is able to reuse log extents that have already been formatted. Whereas linear logging has to allocate new log extents and format them.

See Managing logs for further information.

Advantages of linear logging

The principal advantage of linear logging is that linear logging provides protection against more failures.

Neither circular nor linear logging protect against a corrupted or deleted log, or messages or queues that have been deleted by applications or the administrator.

Linear logging (but not circular) enables damaged objects to be recovered. So, linear logging provides protection against queue files being corrupted or deleted, as these damaged queues can be recovered from a linear log. You can configure automatic media imaging and automatic log management for linear logs to reduce the management overheads. See Automatic media imaging and Automatic log management.

Both circular and linear protect against power loss and communications failure as described in Recovering from power loss or communication failures.

Other considerations

Whether you choose linear or circular depends on how much redundancy you require.

There is a cost to choosing more redundancy, that is linear logging, caused by the performance cost and the administration cost.

See Types of logging for more information.

AIX Shared memory on AIX

If certain application types fail to connect because of an AIX memory limitation, in most cases this can be resolved by setting the environment variable EXTSHM=ON.

Some 32-bit processes on AIX might encounter an operating system limitation that affects their ability to connect to IBM MQ queue managers. Every standard connection to IBM MQ uses shared memory, but unlike other UNIX platforms, AIX allows 32-bit processes to attach only 11 shared memory sets.

Most 32-bit processes will not encounter this limit, but applications with high memory requirements might fail to connect to IBM MQ with reason code 2102: MQRC_RESOURCE_PROBLEM. The following application types might see this error:

- Programs running in 32-bit Java virtual machines
- Programs using the large or very large memory models
- Programs connecting to many queue managers or databases
- Programs that attach to shared memory sets on their own

AIX offers an extended shared memory feature for 32-bit processes that allows them to attach more shared memory. To run an application with this feature, export the environment variable EXTSHM=ON

before starting your queue managers and your program. The EXTSHM=ON feature prevents this error in most cases, but it is incompatible with programs that use the SHM_SIZE option of the shmctl function.

IBM MQ MQI client applications and all 64-bit processes are unaffected by this limitation. They can connect to IBM MQ queue managers regardless of whether EXTSHM has been set.

Linux AIX IBM MQ and UNIX System V IPC resources

A queue manager uses some IPC resources. Use **ipcs -a** to find out what resources are being used.

This information applies to IBM MQ running on AIX and Linux systems only.

IBM MQ uses System V interprocess communication (IPC) resources (*semaphores* and *shared memory segments*) to store and pass data between system components. These resources are used by queue manager processes and applications that connect to the queue manager. IBM MQ MQI clients do not use IPC resources, except for IBM MQ trace control. Use the UNIX command **ipcs -a** to get full information on the number and size of the IPC resources currently in use on the machine.

Linux AIX IBM MQ and UNIX Process Priority

Good practices when setting process priority *nice* values.

This information applies to IBM MQ running on AIX and Linux systems only.

If you run a process in the background, that process can be given a higher *nice* value (and hence lower priority) by the invoking shell. This might have general IBM MQ performance implications. In highly-stressed situations, if there are many ready-to-run threads at a higher priority and some at a lower priority, operating system scheduling characteristics can deprive the lower priority threads of processor time.

It is good practice that independently started processes associated with queue managers, such as **runmqlsr**, have the same *nice* values as the queue manager they are associated with. Ensure the shell does not assign a higher *nice* value to these background processes. For example, in ksh, use the setting "`set +o bgnice`" to stop ksh from raising the *nice* value of background processes. You can verify the *nice* values of running processes by examining the *NI* column of a "ps -efl" listing.

Also, start IBM MQ application processes with the same *nice* value as the queue manager. If they run with different *nice* values, an application thread might block a queue manager thread, or vice versa, causing performance to degrade.

z/OS Planning your IBM MQ environment on z/OS

When planning your IBM MQ environment, you must consider the resource requirements for data sets, page sets, Db2, Coupling Facilities, and the need for logging, and backup facilities. Use this topic to plan the environment where IBM MQ runs.

Before you plan your IBM MQ architecture, familiarize yourself with the basic IBM MQ for z/OS concepts, see the topics in IBM MQ for z/OS concepts.

When planning your queue manager, you might need to work with different people in your organization. It is usually a good idea to involve those people early, as change control procedures can take a long time. They might also be able to tell you what parameters you need to configure IBM MQ for z/OS.

For example you might need to work with the:

- Storage administrator, to determine the high level qualifier of queue manager data sets, and to allocate enough space for queue manager data sets.
- z/OS system programmer to define the IBM MQ subsystem to z/OS and APF authorize the IBM MQ for z/OS libraries.
- Network administrator to determine which TCP/IP stack and ports should be used for IBM MQ for z/OS.
- Security administrator to set up access to queue manager data sets, security profiles for IBM MQ for z/OS resources, and TLS certificates.

- Db2 administrator to set up Db2 tables when configuring a queue sharing group.

Related concepts

IBM MQ Technical overview

Related tasks

“Planning an IBM MQ architecture” on page 5

When planning your IBM MQ environment, consider the support that IBM MQ provides for single and multiple queue manager architectures, and for point-to-point and publish/subscribe messaging styles. Also plan your resource requirements, and your use of logging and backup facilities.

Configuring z/OS

Administering IBM MQ for z/OS

z/OS Planning for your queue manager

When you are setting up a queue manager, your planning should allow for the queue manager to grow, so that the queue manager meets the needs of your enterprise.

The best way to configure a queue manager is in steps:

1. Configure the base queue manager
2. Configure the channel initiator which does queue manager to queue manager communications, and remote client application communication
3. If you want to encrypt and protect messages, configure Advanced Message Security
4. If you want to use File Transfer over IBM MQ, configure Managed File Transfer for z/OS.
5. If you want to use the administrative or messaging REST API, or the IBM MQ Console to manage IBM MQ from a web browser, configure the mqweb server.

Some enterprises have hundreds of thousands of queue managers in their environment. You need to consider your IBM MQ network now, and in five years time.

On z/OS, some queue managers process thousands of messages a second, and log over 100 MB a second. If you expect very high volumes you may need to consider having more than one queue manager.

On z/OS, IBM MQ can run as part of a queue sharing group (QSG) where messages are stored in the Coupling Facility, and any queue manager in the queue sharing group can access the messages. If you want to run in a queue sharing group you need to consider how many queue managers you need. Typically, there is one queue manager for each LPAR. You might also have one queue manager to backup CF structures regularly.

Some changes to configuration are easy to do, such as defining a new queue. Some are harder, such as making logs and page sets bigger; and some configuration cannot be changed, such as the name of a queue manager or the queue sharing group name.

There is performance and tuning information available in the MP16 performance SupportPac .

Naming conventions

You need to have a naming convention for the queue manager data sets.

Many enterprises use the release number in the name of the load libraries, and so on. You might want to consider having an alias of `MQM.SCSQAUTH` pointing to the version currently in use, such as `MQM.V930.SCSQAUTH`, so you do not have to change CICS®, Batch, and IMS JCL when you migrate to a new version of IBM MQ.

You can use a symbolic link in z/OS UNIX System Services to reference the installation directory for the version of IBM MQ currently in use.

The data sets used by the queue manager (logs, page sets, JCL libraries) need a naming convention to simplify the creation of security profiles, and the mapping of data sets to SMS storage classes that control where the data sets are placed on disk, and the attributes they have.

Note, that putting the version of IBM MQ into the name of the page sets or logs, is not a good idea. One day you might migrate to a new version, and the data set will have the "wrong" names.

Applications

You need to understand the business applications and the best way to configure IBM MQ. For example if applications have logic to provide recovery and repeat capability, then non persistent messages might be good enough. If you want IBM MQ to handle the recovery, then you need to use persistent messages and put and get messages in syncpoint.

You need to isolate queues from different business transactions. If a queue for one business application fills up, you do not want this impacting other business applications. Isolate the queues in different page sets and buffer pools, or structures, if possible.

You need to understand the profile of messages. For many applications the queues have only a few messages. Other applications can have queues build up during the day, and be processed overnight. A queue which normally has only a few messages on it, might need to hold many hours worth of messages if there is a problem and messages are not processed. You need to size the CF structures and page sets to allow for your expected peak capacity.

Post configuration

Once you have configured your queue manager (and components) you need to plan for:

- Backing up page sets.
- Backing up definitions of objects.
- Automating the backup of any CF structures.
- Monitoring IBM MQ messages, and taking action when a problem is detected.
- Collecting the IBM MQ statistics data.
- Monitoring resource usage, such as virtual storage, and amount of data logged per hour. With this you can see if your resource usage is increasing and if you need to take actions, such as setting up a new queue manager

z/OS Planning your storage and performance requirements on z/OS

You must set realistic and achievable storage, and performance goals for your IBM MQ system. Use this topic help you understand the factors which affect storage, and performance.

This topic contains information about the storage and performance requirements for IBM MQ for z/OS. It contains the following sections:

- z/OS performance options for IBM MQ
- Determining z/OS workload management importance and velocity goals
- “Library storage” on page 140
- “System LX usage” on page 140
- “Storage configuration” on page 141
- “Disk storage” on page 145

See, “Where to find more information about storage and performance requirements” on page 146 for more information.

z/OS performance options for IBM MQ

With workload management, you define performance goals and assign a business importance to each goal. You define the goals for work in business terms, and the system decides how much resource, such as processor and storage, should be given to the work to meet its goal. Workload management controls the dispatching priority based on the goals you supply. Workload management raises or lowers

the priority as needed to meet the specified goal. Thus, you need not fine-tune the exact priorities of every piece of work in the system and can focus instead on business objectives.

The three kinds of goals are:

Response time
How quickly you want the work to be processed

Execution velocity
How fast the work should be run when ready, without being delayed for processor, storage, I/O access, and queue delay

Discretionary
A category for low priority work for which there are no performance goals

Response time goals are appropriate for end-user applications. For example, CICS users might set workload goals as response time goals. For IBM MQ address spaces, velocity goals are more appropriate. A small amount of the work done in the queue manager is counted toward this velocity goal but this work is critical for performance. Most of the work done by the queue manager counts toward the performance goal of the end-user application. Most of the work done by the channel initiator address space counts toward its own velocity goal. The receiving and sending of IBM MQ messages, which the channel initiator accomplishes, is typically important for the performance of business applications using them.

Determining z/OS workload management importance and velocity goals

See “Determining z/OS workload management importance” on page 140 for more information.

Library storage

You must allocate disk storage for the product libraries. The exact figures depend on your configuration, and should include both the target and distribution libraries, as well as the SMP/E libraries.

The target libraries used by IBM MQ for z/OS use PDSE formats. Ensure that any PDSE target libraries are not shared outside a sysplex. For more information about the required libraries and their sizes and the required format, see the Program Directory. For download links for the Program Directories, see IBM MQ for z/OS Program Directory PDF files.

System LX usage

Each defined IBM MQ subsystem reserves one system linkage index (LX) at IPL time, and a number of non-system linkage indexes when the queue manager is started. The system linkage index is reused when the queue manager is stopped and restarted. Similarly, distributed queuing reserves one non-system linkage index. In the unlikely event of your z/OS system having inadequate system LXs defined, you might need to take these reserved system LXs into account.

If required, the number of system LXs can be increased by setting the *NSYSLX* parameter in SYS1.PARMLIB member IEASYSxx.

z/OS *Determining z/OS workload management importance*

For full information about workload management and defining goals through the service definition, see the z/OS product documentation.

This topic suggests how to set the z/OS workload management importance and velocity goals relative to other important work in your system. See *z/OS MVS Planning: Workload Management* for more information.

The queue manager address space needs to be defined with high priority as it provides subsystem services. The channel initiator is an application address space, but is usually given a high priority to ensure that messages being sent to a remote queue manager are not delayed. Advanced Message Security (AMS) also provides subsystem services and needs to be defined with high priority.

Use the following service classes:

The default SYSSTC service class

- VTAM and TCP/IP address spaces
- IRLM address space (IRLMPROC)

Note: The VTAM, TCP/IP, and IRLM address spaces must have a higher dispatching priority than all the DBMS address spaces, their attached address spaces, and their subordinate address spaces. Do not allow workload management to reduce the priority of VTAM, TCP/IP, or IRLM to (or below) that of the other DBMS address spaces

A high velocity goal and importance of 1 for a service class with a name that you define, such as PRODREGN, for the following:

- IBM MQ queue manager, channel initiator and AMS address spaces
- Db2 (all address spaces, except for the Db2-established stored procedures address space)
- CICS (all region types)
- IMS (all region types except BMPs)

A high velocity goal is good for ensuring that startups and restarts are performed as quickly as possible for all these address spaces.

The velocity goals for CICS and IMS regions are only important during startup or restart. After transactions begin running, workload management ignores the CICS or IMS velocity goals and assigns priorities based on the response time goals of the transactions that are running in the regions. These transaction goals should reflect the relative priority of the business applications they implement. They might typically have an importance value of 2. Any batch applications using IBM MQ should similarly have velocity goals and importance reflecting the relative priority of the business applications they implement. Typically the importance and velocity goals will be less than those for PRODREGN.

z/OS *Storage configuration*

In a 64 bit address space, there is a virtual line called "the bar" that marks the 2GB address. The bar separates storage below the 2GB address, called "below the bar", from storage above the 2GB address, called "above the bar". Storage below the bar uses 31 bit addressability, storage above the bar uses 64 bit addressability.

You can specify the limit of 31-bit storage by using the JCL REGION parameter, and the limit of 64-bit storage by using the MEMLIMIT parameter. These specified values can be overridden by z/OS exits.

Suggested storage configuration

The following table shows suggested **REGION** and **MEMLIMIT** values for the queue manager, channel initiator, and AMS address spaces. These suggestions should be used as a starting point and adjusted using the information in:

- “Queue manager storage configuration” on page 142
- “Channel initiator storage configuration” on page 144

Table 19. Suggested definitions for REGION and MEMLIMIT

Address space	Storage configuration
Queue manager	REGION=0M, MEMLIMIT=3G
Channel initiator from IBM MQ 9.4.0	REGION=0M, MEMLIMIT=2G
AMS address space	REGION=0M

Managing the MEMLIMIT and REGION size

Other mechanisms, for example the **MEMLIMIT** parameter in the SMFPRMxx member of SYS1.PARMLIB or the IEFUSI exit might be used at your installation to provide a default amount of virtual storage above the bar for z/OS address spaces. See Memory management above the bar for full details about limiting storage above the bar.

z/OS *Queue manager storage configuration*

The queue manager address space is likely to be the major user of 64-bit storage in an IBM MQ installation. Each connection to the queue manager requires common storage to be allocated as described in the following text. In addition to 64-bit storage, you should allow the queue manager to use all available 31-bit storage by specifying REGION=0M on the queue manager JCL.

Common storage

Each IBM MQ for z/OS subsystem has the following approximate storage requirements:

- CSA 4KB
- ECSA 800KB, plus the size of the trace table that is specified in the **TRACTBL** parameter of the CSQ6SYSP system parameter macro. For more information, see Using CSQ6SYSP.

In addition, each concurrent logical connection to the queue manager requires about 5 KB of ECSA. When a task ends, other IBM MQ tasks can reuse this storage.

IBM MQ does not release the storage until the queue manager is shut down, so you can calculate the maximum amount of ECSA required by multiplying the maximum number of concurrent connections by 5KB. The number of concurrent logical connections is the sum of the number of:

- Tasks (TCBs) in Batch, TSO, z/OS UNIX System Services, IMS, and Db2 stored procedure address space (SPAS) regions that are connected to IBM MQ, but not disconnected.
- CICS transactions that have issued an IBM MQ request, but have not terminated
- JMS Connections, Sessions, TopicSessions or QueueSessions that have been created (for bindings connection), but not yet destroyed or garbage collected.
- Active IBM MQ channels

You can set a limit to the common storage, used by logical connections to the queue manager, with the **ACELIM** configuration parameter. The **ACELIM** control is primarily of interest to sites where Db2 stored procedures cause operations on IBM MQ queues.

When driven from a stored procedure, each IBM MQ operation can result in a new logical connection to the queue manager. Large Db2 units of work, for example due to table load, can result in an excessive demand for common storage.

ACELIM is intended to limit common storage use and to protect the z/OS system, by limiting the number of connections in the system. You should only set **ACELIM** on queue managers that have been identified as using excessive quantities of ECSA storage. See the **ACELIM** section in *Using CSQ6SYSP* for more information.

To set a value for **ACELIM**, firstly determine the amount of storage currently in the subpool controlled by the **ACELIM** value. This information is in the SMF 115 subtype 5 records produced by statistics CLASS(3) trace.

IBM MQ SMF data can be formatted using SupportPac MP1B. The number of bytes in use in the subpool controlled by **ACELIM** is displayed in the STGPOOL DD, on the line titled *ACE/PEB*.

For more information about SMF 115 statistics records, see Interpreting IBM MQ for z/OS performance statistics.

Increase the normal value by a sufficient margin to provide space for growth and workload spikes. Divide the new value by 1024 to yield a maximum storage size in KB for use in the **ACELIM** configuration.

Private storage

The queue manager address space uses 64-bit storage for many internal control blocks. The **MEMLIMIT** parameter of the queue manager JCL defines the maximum amount of 64-bit storage available. 3GB of storage, MEMLIMIT=3G, is the minimum you should use, however, depending on your configuration significantly more might be required.

You should specify a specific **MEMLIMIT** value rather than MEMLIMIT=NOLIMIT to prevent potential problems. If you specify NOLIMIT or a very large value, then there is the potential to use up all of the available z/OS virtual storage, which leads to paging in your system. When increasing the value of **MEMLIMIT** you should discuss the new setting with your z/OS system programmer in case there is a system-wide limit on the amount of on storage that can be used.

If you have a large value for **MEMLIMIT** you might need to increase the size of your dump data sets as more data is captured in a dump.

If you have a very low value for **MEMLIMIT** you might need to ensure that OUTBUFF is not also near or at its maximum value of 400000, as the queue manager address space requires enough memory for the output buffers.

You can monitor the address space storage usage from the CSQY220I message that indicates the amount of 31 and 64-bit private storage in use, and the remaining free amount.

Buffer pools

Buffer pools are a significant user of private storage in the queue manager address space. Each buffer pool size is determined at queue manager initialization time, and storage is allocated for the buffer pool when a page set that is using that buffer pool is connected. The parameter **LOCATION (ABOVE|BELOW)** is used to specify where the buffers are allocated. You can use the ALTER BUFFPOOL command to dynamically change the size of buffer pools.

When calculating a value for **MEMLIMIT** it is critical that you take into account the buffer pool sizes if they are configured with **LOCATION(ABOVE)**. You should perform the calculation as follows.

Calculate the value of **MEMLIMIT** as 2GB plus the size of the buffer pools configured with **LOCATION(ABOVE)**, rounded up to the nearest GB. Set MEMLIMIT to a minimum of 3GB and increase this as necessary when you need to increase the size of your buffer pools.

For example, for three buffer pools configured with **LOCATION(ABOVE)**, buffer pool one has 10,000 buffers, and buffer pools two and three have 50,000 buffers each. Memory usage above the bar equals 110,000 (total number of buffers) * 4096 = 450,560,000 bytes = 430MB.

All buffer pools regardless of **LOCATION** make use of 64-bit storage for control structures. As the number of buffer pools and number of buffers in those pools increase this can become significant. Each buffer requires around an additional 200 bytes of 64-bit storage. For the preceding configuration that would require: 200 * 110,000 = 22,000,000 bytes = 21MB.

Therefore, in this scenario 3GB can be used for the **MEMLIMIT**, which allows scope for growth: 21MB + 430MB + 2GB which rounds up to 3GB.

For some configurations there can be significant performance benefits to using buffer pools that have their buffers permanently backed by real storage. You can achieve this by specifying the FIXED4KB value for the **PAGECLAS** attribute of the buffer pool. However, you should only do this if there is sufficient real storage available on the LPAR, otherwise other address spaces might be affected. For information about when you should use the FIXED4KB value for **PAGECLAS**, see IBM MQ Support Pac MP16: IBM MQ for z/OS - Capacity planning & tuning.

Making the buffer pools so large that there is MVS paging might adversely affect performance. You might consider using a smaller buffer pool that does not page, with IBM MQ moving the message to and from the page set.

Indexed queues

On z/OS, local queues are indexed if the queue has an **INDXTYPE** attribute that has not been set to NONE. The indexes for shared queues are held in a coupling facility, but for private queues the index is held in 64 bit storage. For each message on an indexed queue 136 bytes of data are used to index the message. For very deep queues this can result in a significant amount of 64 bit storage being allocated. For example, 10 million messages on an indexed queue will use 1.27 GB of 64 bit storage in order to maintain the index.

If you expect to have a large number of messages on indexed queues you should allow for this when setting **MEMLIMIT**. To calculate an upper limit for the amount of storage required for indexes, multiply the **MAXDEPTH** attribute for each indexed queue by 136 and sum the value. This value should be added to your existing **MEMLIMIT**.

RECOVER CFSTRUCT

From IBM MQ 9.4.0 the **RECOVER CFSTRUCT** command makes greater use of 64-bit storage. In many cases there should be spare 64-bit storage available and so use of the command does not require an increase in the value of **MEMLIMIT**. However, if you are likely to have large structure backups, containing more than a few million messages, you should increase the **MEMLIMIT** for all queue managers which might process the **RECOVER CFSTRUCT** command by 500MB.

For example if you had MEMLIMIT=3G already, you should consider using MEMLIMIT=4G as the **MEMLIMIT** parameter does not allow for decimal points.

Shared Message Data Set (SMDS) buffers and MEMLIMIT

When running messaging workloads using shared message data sets, there are two levels of optimizations that can be achieved by adjusting the DSBUFS and DSBLOCK attributes.

The amount of above bar queue manager storage used by the SMDS buffer is DSBUFS x DSBLOCK. This means that by default, 100 x 256KB (25MB) is used for each CFLEVEL(5) structure in the queue manager.

Although this value is not too high, if your enterprise, or enterprises have many CFSTRUCTs, some of them might allocate a high value of MEMLIMIT for buffer pools, and sometimes they have deep indexed queues, so in total, they might run out of storage above the bar.

z/OS *Channel initiator storage configuration*

The channel initiator typically uses much less 64-bit storage than the queue manager. However, from IBM MQ 9.4.0 the usage has increased. In addition to 64-bit storage, you should allow the channel initiator to use all available 31-bit storage by specifying REGION=0M on the channel initiator JCL.

Common storage

The channel initiator typically requires ECSA usage of up to 160KB.

31-bit private storage

The 31-bit storage available to the channel initiator limits the number of concurrent connections the CHINIT can have.

Every channel uses approximately 170KB of extended private region in the channel initiator address space. For message channels, for example, sender or receiver channels, storage is increased by message size if messages larger than 32KB are transmitted. This increased storage is freed when:

- A sending or client channel requires less than half the current buffer size for 10 consecutive messages.
- A heartbeat is sent or received.

The storage is freed for reuse within the Language Environment, however, the storage is not seen as free by the z/OS virtual storage manager. This means that the upper limit for the number of channels is dependent on message size and arrival patterns, and on limitations of individual user systems on extended private region size.

The upper limit on the number of channels is likely to be approximately 9000 on many systems because the extended region size is unlikely to exceed 1.6GB.

The channel initiator trace is written to a data space. The size of the data space storage, is controlled by the **TRAXTBL** parameter. See ALTER QMGR.

64-bit private storage

The **MEMLIMIT** parameter of the channel initiator JCL defines the maximum amount of 64-bit storage available. 2 GB of storage, `MEMLIMIT=2G`, is the minimum value you should use. Depending on your configuration significantly more might be required.

You should specify a sensible **MEMLIMIT** value rather than `MEMLIMIT=NOLIMIT` to prevent potential problems. If you specify `NOLIMIT` or a very large value, then there is the potential to use up all of the available z/OS virtual storage, leading to paging in your system. When increasing the value of **MEMLIMIT** you should discuss the new setting with your z/OS system programmer in case there is a system-wide limit on the amount of on storage that can be used.

If you have a large value for **MEMLIMIT** you might need to increase the size of your dump data sets as more data is captured in a dump.

There are two users of 64-bit storage in the channel initiator: SMF and server-connection channels.

SMF

If enabled, SMF class 4 accounting, or statistics, require 64-bit storage. A minimum of 256MB storage is required. If sufficient storage is not available, the channel initiator issues the CSQX124E message and class 4 accounting and statistics are not available.

Server-connection channels

From IBM MQ 9.4.0 server-connection channels allocate message buffers in 64-bit storage, if they are transferring messages larger than 32 KB in size.

These buffers are freed if the channels require less than half the current buffer size for 10 consecutive messages, or a heartbeat is sent or received.

The value of **MEMLIMIT** sets an upper limit on how many concurrent server-connection channels can run. You should use a minimum value of `MEMLIMIT=2G` to ensure that the same number of channels can run as in earlier versions of IBM MQ, as well as providing some capacity for growth.

You can calculate an approximate value for **MEMLIMIT** by working out the peak maximum number of concurrently active server-connection channels, and for those channels the maximum message size you expect them to transfer. You should use `MEMLIMIT=2G` as a starting point and round up.

For example, if you set the maximum number of concurrent server-connection channels to be 2,000 and each channel to have a maximum message size of 1MB, then server-connection channels are using a maximum of just under 2GB of 64-bit storage. As this is very close to 2GB then you should round up to `MEMLIMIT=3G`.

z/OS *Disk storage*

Use this topic when planning your disk storage requirements for log data sets, Db2 storage, coupling facility storage, and page data sets.

Work with your storage administrator to determine where to put the queue manager data sets. For example, your storage administrator may give you specific DASD volumes, or SMS storage classes, data classes, and management classes for the different data set types.

- Log data sets must be on DASD. These logs can have high I/O activity with a small response time and do not need to be backed up.

- Archive logs can be on DASD or tape. After they have been created, they might never be read again except in an abnormal situation, such as recovering a page set from a backup. They should have a long retention date.
- Page sets might have low to medium activity and should be backed up regularly. On a high use system, they should be backed up twice a day.
- BSDS data sets should be backed up daily; they do not have high I/O activity.

All data sets are similar to those used by Db2, and similar maintenance procedures can be used for IBM MQ.

See the following sections for details of how to plan your data storage:

- **Logs and archive storage**

  "How long do I need to keep archive logs" on page 165 describes how to determine how much storage your active log and archive data sets require, depending on the volume of messages that your IBM MQ system handles and how often the active logs are offloaded to your archive data sets.
- **Db2 storage**

  "Db2 storage" on page 182 describes how to determine how much storage Db2 requires for the IBM MQ data.
- **coupling facility storage**

  "Defining coupling facility resources" on page 172 describes how to determine how large to make your coupling facility structures.
- **Page set and message storage**

  "Planning your page sets and buffer pools" on page 147 describes how to determine how much storage your page data sets require, depending on the sizes of the messages that your applications exchange, on the numbers of these messages, and on the rate at which they are created or exchanged.

z/OS *Where to find more information about storage and performance requirements*

Use this topic as a reference to find more information about storage and performance requirements.

You can find more information from the following sources:

Table 20. Where to find more information about storage requirements

Topic	Where to look
System parameters	Using CSQ6SYSP and Customizing your queue managers
Storage required to install IBM MQ	Program Directory. For download links for the Program Directories, see IBM MQ for z/OS Program Directory PDF files.
IEALIMIT and IEFUSI exits	See IEALIMIT and IEFUSI in the *z/OS:MVS Installation Exits* documentation.
Latest information	IBM MQ SupportPac website SupportPacs for IBM MQ and other project areas.
Workload management and defining goals through the service definition	*z/OS MVS Planning: Workload Management*

z/OS Planning your page sets and buffer pools

Information to help you with planning the initial number, and sizes of your page data sets, and buffer pools.

This topic contains the following sections:

- "Plan your page sets" on page 147
  - Page set usage
  - Number of page sets
  - Size of page sets
  - Planning for z/OS data set encryption
- "Calculate the size of your page sets" on page 148
  - Page set zero
  - Page set 01 - 99
  - Calculating the storage requirement for messages
- "Enabling dynamic page set expansion" on page 150
- "Defining your buffer pools" on page 152

Plan your page sets

Page set usage

For short-lived messages, few pages are normally used on the page set and there is little or no I/O to the data sets except at startup, during a checkpoint, or at shutdown.

For long-lived messages, those pages containing messages are normally written out to disk. This operation is performed by the queue manager in order to reduce restart time.

Separate short-lived messages from long-lived messages by placing them on different page sets and in different buffer pools.

Number of page sets

Using several large page sets can make the role of the IBM MQ administrator easier because it means that you need fewer page sets, making the mapping of queues to page sets simpler.

Using multiple, smaller page sets has a number of advantages. For example, they take less time to back up, and I/O can be carried out in parallel during backup and restart. However, consider that this adds a significant performance cost to the role of the IBM MQ administrator, who is required to map each queue to one of a much greater number of page sets.

Define at least five page sets, as follows:

- A page set reserved for object definitions (page set zero)
- A page set for system-related messages
- A page set for performance-critical long-lived messages
- A page set for performance-critical short-lived messages
- A page set for all other messages

"Defining your buffer pools" on page 152 explains the performance advantages of distributing your messages on page sets in this way.

Size of page sets

Define sufficient space in your page sets for the expected peak message capacity. Consider for any unexpected peak capacity, such as when a build-up of messages develops because a queue server program is not running. You can do this by allocating the page set with secondary extents or,

alternatively, by enabling dynamic page set expansion. For more information, see “Enabling dynamic page set expansion” on page 150. It is difficult to make a page set smaller, so it is often better to allocate a smaller page set, and allow it to expand when needed.

When planning page set sizes, consider all messages that might be generated, including non-application message data. For example, trigger messages, event messages and any report messages that your application has requested.

The size of the page set determines the time taken to recover a page set when restoring from a backup, because a large page set takes longer to restore.

Note: Recovery of a page set also depends on the time the queue manager takes to process the log records written since the backup was taken; this time period is determined by the backup frequency. For more information, see “Planning for backup and recovery” on page 184.

Note: Page sets larger than 4 GB require the use of SMS extended addressability.

Planning for z/OS data set encryption

You can apply the z/OS data set encryption feature to page sets for queue managers running at IBM MQ for z/OS 9.1.4 or later.

You must allocate these page sets with EXTENDED attributes, and a data set key label that ensures the data is AES encrypted.

See the section, confidentiality for data at rest on IBM MQ for z/OS with data set encryption. for more information.

Calculate the size of your page sets

For queue manager object definitions (for example, queues and processes), it is simple to calculate the storage requirement because these objects are of fixed size and are permanent. For messages however, the calculation is more complex for the following reasons:

- Messages vary in size.
- Messages are transitory.
- Space occupied by messages that have been retrieved is reclaimed periodically by an asynchronous process.

Large page sets of greater than 4 GB that provide extra capacity for messages if the network stops, can be created if required. It is not possible to modify the existing page sets. Instead, new page sets with extended addressability and extended format attributes, must be created. The new page sets must be the same physical size as the old ones, and the old page sets must then be copied to the new ones.
If backward migration is required, page set zero must not be changed. If page sets less than 4 GB are adequate, no action is needed.

Page set zero

Page set zero is reserved for object definitions.

V 10.0.0 For page set zero, the storage required is approximately:

```
    (maximum number of local queue definitions x 1364)
       (excluding shared queues)
+   (maximum number of model queue definitions x 1024)
+   (maximum number of alias queue definitions x 512)
+   (maximum number of remote queue definitions x 685)
+   (maximum number of permanent dynamic queue definitions x 1364)
+   (maximum number of process definitions x 680)
+   (maximum number of namelist definitions x 12320)
+   (maximum number of message channel definitions x 2026)
+   (maximum number of client-connection channel definitions x 5170)
+   (maximum number of server-connection channel definitions x 2048)
+   (maximum number of storage class definitions x 287)
+   (maximum number of authentication information definitions x 1010)
+   (maximum number of administrative topic definitions x 15000)
+   (total length of topic strings defined in administrative topic definitions)
```

V 10.0.0 Before IBM MQ 10.0.0, message channel definitions always use the same amount of space. From IBM MQ 10.0.0, the amount of space used varies depending on the exact channel definition. 2026 bytes is a good value for storage planning purposes.

Divide this value by 4096 to determine the number of records to specify in the cluster for the page set data set.

You do not need to allow for objects that are stored in the shared repository, but you must allow for objects that are stored or copied to page set zero (objects with a disposition of GROUP or QMGR).

The total number of objects that you can create is limited by the capacity of page set zero. The number of local queues that you can define is limited to 524 287.

Page sets 01 - 99

For page sets 01 - 99, the storage required for each page set is determined by the number and size of the messages stored on that page set. (Messages on shared queues are not stored on page sets.)

Divide this value by 4096 to determine the number of records to specify in the cluster for the page set data set.

Calculating the storage requirement for messages

This section describes how messages are stored on pages. Understanding this can help you calculate how much page set storage you must define for your messages. To calculate the approximate space required for all messages on a page set you must consider maximum queue depth of all the queues that map to the page set and the average size of messages on those queues.

Note: The sizes of the structures and control information given in this section are liable to change between major releases. For details specific to your release of IBM MQ, refer to SupportPac MP16 - IBM MQ for z/OS Capacity planning & tuning and IBM MQ Family - Performance Reports

You must allow for the possibility that message "gets" might be delayed for reasons outside the control of IBM MQ (for example, because of a problem with your communications protocol). In this case, the "put" rate of messages might far exceed the "get" rate. This can lead to a large increase in the number of messages stored in the page sets and a consequent increase in the storage size demanded.

Each page in the page set is 4096 bytes long. Allowing for fixed header information, each page has 4057 bytes of space available for storing messages.

When calculating the space required for each message, the first thing you must consider is whether the message fits on one page (a short message) or whether it needs to be split over two or more pages (a long message). When messages are split in this way, you must allow for additional control information in your space calculations.

For the purposes of space calculation, a message can be represented as the following:

The message header section contains the message descriptor and other control information, the size of which varies depending on the size of the message. The message data section contains all the actual message data, and any other headers (for example, the transmission header or the IMS bridge header).

A minimum of two pages are required for page set control information which, is typically less than 1% of the total space required for messages.

Short messages

A short message is defined as a message that fits on one page.

Small messages are stored one on each page.

Long messages

If the size of the message data is greater than 3596 bytes, but not greater than 4 MB, the message is classed as a long message. When presented with a long message, IBM MQ stores the message on a series of pages, and stores control information that points to these pages in the same way that it would store a short message. This is shown in Figure 41 on page 150:

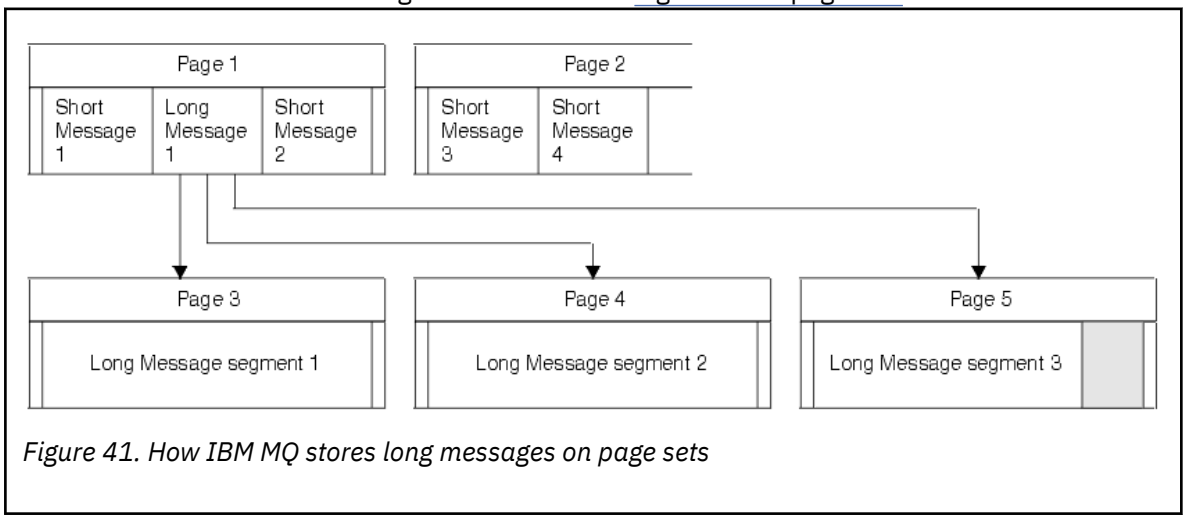


Figure 41. How IBM MQ stores long messages on page sets

Very long messages

Very long messages are messages with a size greater than 4 MB. These are stored so that each 4 MB uses 1037 pages. Any remainder is stored in the same way as a long message, as described above.

z/OS *Enabling dynamic page set expansion*

Page sets can be extended dynamically while the queue manager is running. A page set can have 123 extents, and can be spread over multiple disk volumes.

Each time a page set expands, a new data set extent is used. The queue manager continues to expand a page set when required, until the maximum number of extents has been reached, or until no more storage is available for allocation on eligible volumes.

Once page set expansion fails for one of the reasons above, the queue manager marks the page set for no further expansion attempts. This marking can be reset by altering the page set to EXPAND(SYSTEM).

Page set expansion takes place asynchronously to all other page set activity, when 90% of the existing space in the page set is allocated.

The page set expansion process formats the newly allocated extent and makes it available for use by the queue manager. However, none of the space is available for use, until the entire extent has been formatted. This means that expansion by a large extent is likely to take some time, and putting applications might 'block' if they fill the remaining 10% of the page set before the expansion has completed.

Sample thlqual.SCSQPROC(CSQ4PAGE) shows how to define the secondary extents.

To control the size of new extents, you use one of the following options of the EXPAND keyword of the DEFINE PSID and ALTER PSID commands:

- USER
- SYSTEM
- NONE

USER

Uses the secondary extent size specified when the page set was allocated. If a value was not specified, or if a value of zero was specified, dynamic page set expansion cannot occur.

Page set expansion occurs when the space in the page is 90% used, and is performed asynchronously with other page set activity.

This may lead to expansion by more than a single extent at a time.

Consider the following example: you allocate a page set with a primary extent of 100,000 pages and a secondary extent of 5000 pages. A message is put that requires 9999 pages. If the page set is already using 85,000 pages, writing the message crosses the 90% full boundary (90,000 pages). At this point, a further secondary extent is allocated to the primary extent of 100,000 pages, taking the page set size to 105,000 pages. The remaining 4999 pages of the message continue to be written. When the used page space reaches 94,500 pages, which is 90% of the updated page set size of 105,000 pages, another 5000 page extent is allocated, taking the page set size to 110,000 pages. At the end of the MQPUT, the page set has expanded twice, and 94,500 pages are used. None of the pages in the second page set expansion have been used, although they were allocated.

At restart, if a previously used page set has been replaced with a data set that is smaller, it is expanded until it reaches the size of the previously used data set. Only one extent is required to reach this size.

SYSTEM

Ignores the secondary extent size that was specified when the page set was defined. Instead, the queue manager sets a value that is approximately 10% of the current page set size. The value is rounded up to the nearest cylinder of DASD.

If a value was not specified, or if a value of zero was specified, dynamic page set expansion can still occur. The queue manager sets a value that is approximately 10% of the current page set size. The new value is rounded up depending on the characteristics of the DASD.

Page set expansion occurs when the space in the page set is approximately 90% used, and is performed asynchronously with other page set activity.

At restart, if a previously used page set has been replaced with a data set that is smaller, it is expanded until it reaches the size of the previously used data set.

NONE

No further page set expansion is to take place.

Related reference

ALTER PSID

DEFINE PSID

DISPLAYUSAGE

Defining your buffer pools

Use this topic to help plan the number of buffer pools you should define, and their settings.

This topic is divided into the following sections:

1. "Decide on the number of buffer pools to define" on page 152
2. "Decide on the settings for each buffer pool" on page 153
3. "Monitor the performance of buffer pools under expected load" on page 153
4. "Adjust buffer pool characteristics" on page 153

Decide on the number of buffer pools to define

You should define four buffer pools initially:

Buffer pool 0

Use for object definitions (in page set zero) and performance critical, system related message queues, such as the SYSTEM.CHANNEL.SYNCQ queue and the SYSTEM.CLUSTER.COMMAND.QUEUE and SYSTEM.CLUSTER.REPOSITORY.QUEUE queues.

However it is important to consider point "7" on page 154 in *Adjust buffer pool characteristics* if a large number of channels, or clustering, is to be used.

Use the remaining three buffer pools for user messages.

Buffer pool 1

Use for important long-lived messages.

Long-lived messages are those that remain in the system for longer than two checkpoints, at which time they are written out to the page set. If you have many long-lived messages, this buffer pool should be relatively small, so that page set I/O is evenly distributed (older messages are written out to DASD each time the buffer pool becomes 85% full).

If the buffer pool is too large, and the buffer pool never gets to 85% full, page set I/O is delayed until checkpoint processing. This might affect response times throughout the system.

If you expect few long-lived messages only, define this buffer pool so that it is sufficiently large to hold all these messages.

Buffer pool 2

Use for performance-critical, short-lived messages.

There is normally a high degree of buffer reuse, using few buffers. However, you should make this buffer pool large to allow for unexpected message accumulation, for example, when a server application fails.

Buffer pool 3

Use for all other (typically, performance noncritical) messages.

Queues such as the dead-letter queue, SYSTEM.COMMAND.* queues and SYSTEM.ADMIN.* queues can also be mapped to buffer pool 3.

Where virtual storage constraints exist, and buffer pools need to be smaller, buffer pool 3 is the first candidate for size reduction.

You might need to define additional buffer pools in the following circumstances:

- If a particular queue is known to require isolation, perhaps because it exhibits different behavior at various times.
  - Such a queue might either require the best performance possible under the varying circumstances, or need to be isolated so that it does not adversely affect the other queues in a buffer pool.
  - Each such queue can be isolated into its own buffer pool and page set.
- You want to isolate different sets of queues from each other for class-of-service reasons.

– Each set of queues might then require one, or both, of the two types of buffer pools 1 or 2, as described in Suggested definitions for buffer pool settings, necessitating creation of several buffer pools of a specific type.

Decide on the settings for each buffer pool

If you are using the four buffer pools described in “Decide on the number of buffer pools to define” on page 152, then Suggested definitions for buffer pool settings gives two sets of values for the size of the buffer pools.

The first set is suitable for a test system, the other for a production system or a system that will become a production system eventually. In all cases define your buffer pools with the **LOCATION**(*ABOVE*) attribute

Table 21. Suggested definitions for buffer pool settings

Definition setting	Test system	Production system
BUFFPOOL 0	1 050 buffers	50 000 buffers
BUFFPOOL 1	1 050 buffers	20 000 buffers
BUFFPOOL 2	1 050 buffers	50 000 buffers
BUFFPOOL 3	1 050 buffers	20 000 buffers

If you need more than the four suggested buffer pools, select the buffer pool (1 or 2) that most accurately describes the expected behavior of the queues in the buffer pool, and size it using the information in Suggested definitions for buffer pool settings.

Ensure that your MEMLIMIT is set high enough, so that all the buffer pools can be located above the bar.

Monitor the performance of buffer pools under expected load

You can monitor the usage of buffer pools by analyzing buffer pool performance statistics. In particular, you should ensure that the buffer pools are large enough so that the values of QPSTSOS, QPSTSTLA, and QPSTDMC remain at zero.

For further information, see Buffer manager data records.

Adjust buffer pool characteristics

Use the following points to adjust the buffer pool settings from “Decide on the settings for each buffer pool” on page 153, if required.

Use the performance statistics from “Monitor the performance of buffer pools under expected load” on page 153 as guidance.

1. If you are migrating from an earlier version of IBM MQ, only change your existing settings if you have more real storage available.
2. In general, bigger buffer pools are better for performance, and buffer pools can be much bigger if they are above the bar.

   However, at all times you should have sufficient real storage available so that the buffer pools are resident in real storage. It is better to have smaller buffer pools that do not result in paging, than big ones that do.

   Additionally, there is no point having a buffer pool that is bigger than the total size of the page sets that use it, although you should take into account page set expansion if it is likely to occur.
3. Aim for one page set per buffer pool, as this provides better application isolation.
4. If you have sufficient real storage, such that your buffer pools will never be paged out by the operating system, consider using page-fixed buffers in your buffer pool.

This is particularly important if the buffer pool is likely to undergo much I/O, as it saves the CPU cost associated with page-fixing the buffers before the I/O, and page-unfixing them afterwards.

5. There are several benefits to locating buffer pools above the bar even if they are small enough to fit below the bar. These are:
   - 31 bit virtual storage constraint relief - for example more space for common storage.
   - If the size of a buffer pool needs to be increased unexpectedly while it is being heavily used, there is less impact and risk to the queue manager, and its workload, by adding more buffers to a buffer pool that is already above the bar, than moving the buffer pool to above the bar and then adding more buffers.
6. Tune buffer pool zero and the buffer pool for short-lived messages (buffer pool 2) so that the 15% free threshold is never exceeded (that is, QPSTCBSL divided by QPSTNBUF is always greater than 15%). If more than 15% of buffers remain free, I/O to the page sets using these buffer pools can be largely avoided during normal operation, although messages older than two checkpoints are written to page sets.

   Attention: The optimum value for these parameters is dependent on the characteristics of the individual system. The values given are intended only as a guideline and might not be appropriate for your system.
7. SYSTEM.* queues which get very deep, for example SYSTEM.CHANNEL.SYNCQ, might benefit from being placed in their own buffer pool, if sufficient storage is available.

IBM MQ SupportPac MP16 - IBM MQ for z/OS Capacity planning & tuning provides further information about tuning buffer pools.

z/OS Planning your logging environment on z/OS

Use this topic to plan the number, size and placement of the logs, and log archives used by IBM MQ.

Logs are used to:

- Write recovery information about persistent messages
- Record information about units of work using persistent messages
- Record information about changes to objects, such as define queue
- Backup CF structures

and for other internal information.

The IBM MQ logging environment is established using the system parameter macros to specify options, such as: whether to have single or dual active logs, what media to use for the archive log volumes, and how many log buffers to have.

These macros are described in Create the bootstrap and log data sets and Tailor your system parameter module.

Note: If you are using queue sharing groups, ensure that you define the bootstrap and log data sets with SHAREOPTIONS(2 3).

This section contains information about the following topics:

z/OS *Log data set definitions*

Use this topic to decide on the most appropriate configuration for your log data sets.

This topic contains information to help you answer the following questions:

- Should your installation use single or dual logging?
- How many active log data sets do you need?
- “How large should the active logs be?” on page 156
- Active log placement

- “Active log encryption with z/OS data set encryption” on page 157

Should your installation use single or dual logging?

In general you should use dual logging for production, to minimize the risk of losing data. If you want your test system to reflect production, both should use dual logging, otherwise your test systems can use single logging.

With single logging data is written to one set of log data sets. With dual logging data is written to two sets of log data sets, so in the event of a problem with one log data set, such as the data set being accidentally deleted, the equivalent data set in the other set of logs can be used to recover the data.

With dual logging you require twice as much DASD as with single logging.

If you are using dual logging, then also use dual BSDSs and dual archiving to ensure adequate provision for data recovery.

Dual active logging adds a small performance cost.

Attention: Use of disk mirroring technologies, such as Metro Mirror, are not necessarily a replacement for dual logging and dual BSDS. If a mirrored data set is accidentally deleted, both copies are lost.

If you use persistent messages, single logging can increase maximum capacity by 10-30% and can also improve response times.

Single logging uses 2 - 310 active log data sets, whereas dual logging uses 4 - 620 active log data sets to provide the same number of active logs. Thus single logging reduces the amount of data logged, which might be important if your installation is I/O constrained.

How many active log data sets do you need?

The number of logs depends on the activities of your queue manager. For a test system with low throughput, three active log data sets might be suitable. For a high throughput production system you might want the maximum number of logs available, so, if there is a problem with offloading logs you have more time to resolve the problems.

You must have at least three active log data sets, but it is preferable to define more. For example, if the time taken to fill a log is likely to approach the time taken to archive a log during peak load, define more logs.

Note: Page sets and active log data sets are eligible to reside in the extended addressing space (EAS) part of an extended address volumes (EAV) and an archive log dataset can also reside in the EAS.

You should also define more logs to offset possible delays in log archiving. If you use archive logs on tape, allow for the time required to mount the tape.

Consider having enough active log space to keep a day's worth of data, in case the system is unable to archive because of lack of DASD or because it cannot write to tape. If all the active logs fill up, then IBM MQ is unable to process persistent messages or transactions. It is very important to have enough active log space.

It is possible to dynamically define new active log data sets as a way of minimizing the effect of archive delays or problems. New data sets can be brought online rapidly, using the **DEFINE LOG** command to avoid queue manager 'stall' due to lack of space in the active log.

If you want to define more than 31 active log data sets, you must configure your logging environment to use a version 2 format BSDS. Once a version 2 format BSDS is in use, up to 310 active log data sets can be defined for each log copy ring. See “Planning to increase the maximum addressable log range” on page 166 for information on how you convert to a version 2 format BSDS.

You can tell whether your queue manager is using a version 2 or higher BSDS, either by running the print log map utility (CSQJU004), or from the CSQJ034I message issued during queue manager initialization.

An end of log RBA range of FFFFFFFFFFFFFFFF, in the CSQJ034I message, indicates that a version 2, or higher, format BSDS is in use. An end of log RBA range of 0000FFFFFFFFFFFF, in the CSQJ034I message, indicates that a version 1 format BSDS is in use.

When a queue manager is using a version 2, or higher, format BSDS, it is possible to use the **DEFINE LOG** command to dynamically add more than 31 active log data sets to a log copy ring.

How large should the active logs be?

The maximum supported active log size, when archiving to disk or to tape, is 4 GB.

You should create active logs of at least 1 GB in size for production and test systems.

Important: You need to be careful when allocating data sets, because IDCAMS rounds up the size you allocate.

To allocate a 3 GB log specify one of the following options:

- Cylinders(4369)
- Megabytes(3071)
- TRACKS(65535)
- RECORD(786420)

Any one of these allocates 2.99995 GB.

To allocate a 4GB log specify one of the following options:

- Cylinders(5825)
- Megabytes(4095)
- TRACKS(87375)
- RECORD(1048500)

Any one of these allocates 3.9997 GB.

When using striped data sets, where the data set is spread across multiple volumes, the specified size value is allocated on each DASD volume used for striping. So, if you want to use 4 GB logs and four volumes for striping, you should specify:

- CYLinders(1456)
- Megabytes(1023)

Setting these attributes allocates 4*1456 = 5824 Cylinders or 4 * 1023 = 4092 Megabytes.

Note: Striping is supported when using extended format data sets. This is usually set by the storage manager.

See Increasing the size of the active log for information on carrying out the procedure.

Active log placement

You should work with your storage management team to set up storage pools for the queue managers. You need to consider:

- A naming convention, so the queue managers use the correct SMS definitions.
- Space required for active and archive logs. Your storage pool should have enough space for the active logs from a whole day.
- Performance and resilience to failures.

For performance reasons you should consider striping your active log data sets. The I/O is spread across multiple volumes and reduces the I/O response times, leading to higher throughput. See the preceding text for information about allocating the size of the active logs when using striping.

You should review the I/O statistics using reports from RMF or a similar product. Perform the review of these statistics monthly (or more frequently) for the IBM MQ data sets, to ensure there are no delays due to the location of the data sets.

In some situations, there can be much IBM MQ page set I/O, and this can impact the IBM MQ log performance if they are located on the same DASD.

If you use dual logging, ensure that each set of active and archive logs is kept apart. For example, allocate them on separate DASD subsystems, or on different devices.

This reduces the risk of them both being lost if one of the volumes is corrupted or destroyed. If both copies of the log are lost, the probability of data loss is high.

When you create a new active log data, set you should preformat it using CSQJUFMT. If the log is not preformatted, the queue manager formats the log the first time it is used, which impacts the performance.

With older DASD with large spinning disks, you had to be careful which volumes were used to get the best performance.

With modern DASD, where data is spread over many PC sized disks, you do not need to worry so much about which volumes are used.

Your storage manager should be checking the enterprise DASD to review and resolve any performance problems. For availability, you might want to use one set of logs on one DASD subsystem, and the dual logs on a different DASD subsystem.

Active log encryption with z/OS data set encryption

You can apply the z/OS data set encryption feature to active log data sets for queue managers running at IBM MQ for z/OS 9.1.4 or later.

You must allocate these active log data sets with EXTENDED attributes, and a data set key label that ensures the data is AES encrypted.

See the section, confidentiality for data at rest on IBM MQ for z/OS with data set encryption. for more information.

z/OS *Using MetroMirror with IBM MQ*

IBM Metro Mirror, previously known as Synchronous Peer to Peer Remote Copy (PPRC), is a synchronous replication solution between two storage subsystems, where write operations are completed on both the primary and secondary volumes before the write operation is considered to be complete. Metro Mirror can be used in environments that require no data loss in the event of a storage subsystem failure.

Supported data set types

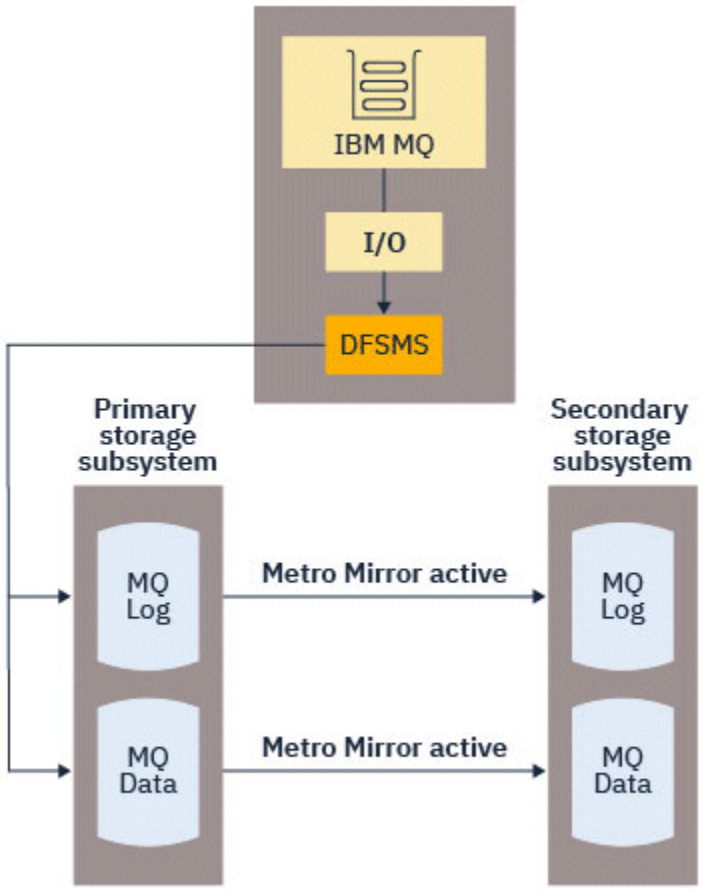
All of the following IBM MQ data set types can be replicated using Metro Mirror. However, exactly which ones are replicated depends on the availability requirements of your enterprise:

- Active logs
- Archive logs
- Bootstrap data set (BSDS)
- Page sets
- Shared message data set (SMDS)
- Data sets used for configuration, for example, in the CSQINP* DD cards on the MSTR JCL

Using zHyperWrite with IBM MQ active logs

When a write is made to a data set that is replicated using Metro Mirror, the write is first made to the primary volume, and then replicated to the secondary volume. This replication is done by the storage subsystem and is transparent to the application that issued the write, for example IBM MQ.

This process is illustrated in the following diagram.

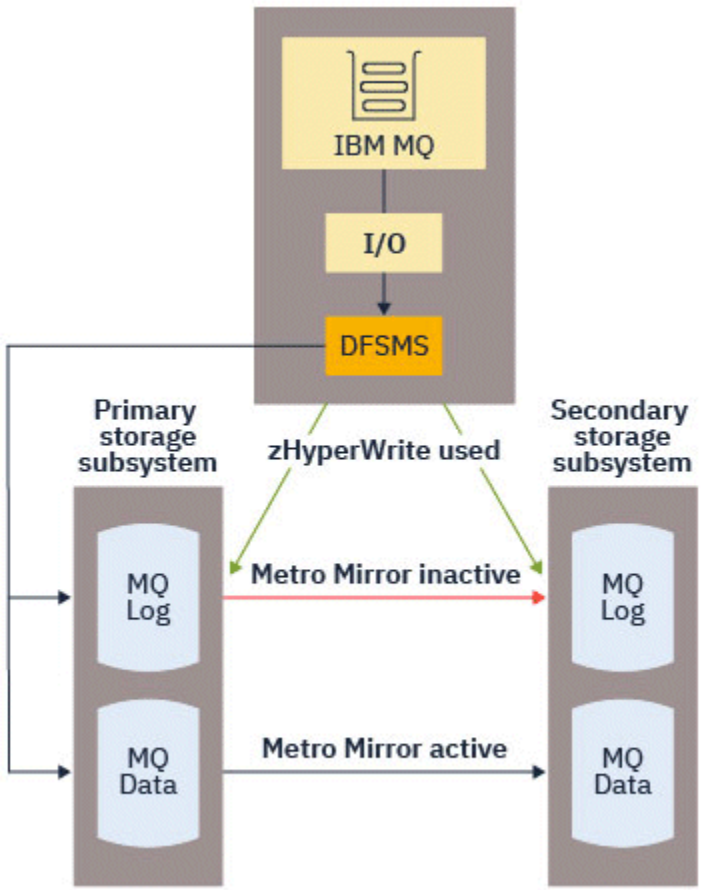


Because both writes to the primary and secondary storage subsystems need to complete before the write returns to IBM MQ, use of Metro Mirror can have a performance impact. You need to balance this performance impact against the availability benefits of using Metro Mirror.

The IBM MQ active logs are most sensitive to the performance impact of using Metro Mirror. IBM MQ allows use of zHyperWrite with the active logs to help reduce this performance impact.

zHyperWrite is a storage subsystem technology that works with z/OS to reduce the performance impact of writes made to data sets that are replicated using Metro Mirror. When zHyperWrite is used, the write to the primary and secondary volumes are issued in parallel at the Data Facility Storage Management Subsystem (DFSMS) level, instead of sequentially at the storage subsystem level, thereby reducing the performance impact.

The following diagram illustrates zHyperWrite being used for the active logs, and Metro Mirror being used for the other IBM MQ data set types. Note that if a zHyperWrite write fails, DFSMS will transparently reissue the write using Metro Mirror.



zHyperWrite on IBM MQ, is supported only on the active log data sets.

In order to use zHyperWrite with the active logs, you need to:

- Configure IBM MQ to use zHyperWrite, and
- The active logs need to be on zHyperWrite capable volumes

You can configure IBM MQ to use zHyperWrite by using one of the following methods:

- Specify ZHYWRITE(YES) in the system parameter module.
- Issue the command SET LOG ZHYWRITE(YES).

Set the following conditions for active log data sets to be on zHyperWrite capable volumes:

- Enable the volumes for Metro Mirror, and the volumes support zHyperWrite
- Ensure that the volumes are HyperSwap enabled
- Specify HYPERWRITE=YES in the IECIOSxx parameter

Prior to IBM MQ 9.4.0, if all the preceding conditions are met, then writes to the active logs are enabled for zHyperWrite. If one, or more, of these conditions are not met, IBM MQ writes to the active logs as normal, and Metro Mirror replicates the writes if it is configured.

From IBM MQ 9.4.0, if ZHYWRITE(YES) is specified, then IBM MQ always attempts to use zHyperWrite when writing to the active logs, regardless of whether the logs are on zHyperWrite capable volumes. If the logs are not on zHyperWrite capable volumes then Metro Mirror replicates the writes if it is configured. There are no negative effects of attempting to use zHyperWrite if the logs are not on zHyperWrite capable volumes

Notes:

- IBM MQ does not require that all active log data sets are on zHyperWrite capable volumes.

If IBM MQ detects that some active log data sets are on zHyperWrite capable volumes, and others are not, it issues message CSQJ166E and carries on processing.

- IBM MQ checks whether active log data sets are zHyperWrite capable when the data sets are first opened.

  Log data sets are opened either at queue manager start up, or when dynamically adding using the DEFINE LOG command. If the log data sets are made zHyperWrite capable while a queue manager has them open, the queue manager will not detect this until it has been restarted.

You can use the output of the DISPLAY LOG command to indicate whether the current active log data sets are zHyperWrite capable. The following example shows that both of the data sets are zHyperWrite capable. If the queue manager has been configured with ZHYWRITE(YES), writes to these logs would be enabled for zHyperWrite:

```
Copy %Full zHyperWrite DSName
 1      4  CAPABLE     MQTST.SUBSYS.MQDL.LOGCOPY1.DS001
 2      4  CAPABLE     MQTST.SUBSYS.MQDL.LOGCOPY2.DS001
```

z/OS ***Faster log throughput with zHyperLink***

zHyperLink technology is designed to reduce input/output (I/O) latency by providing a fast, reliable, and direct communication path between the CPU and the I/O device.

Overview of zHyperLink

zHyperLink can improve active log throughput and reduce IBM MQ transaction time by up to 3.5 times. This goal is accomplished by installing zHyperLink adapters on the z/OS host, select IBM storage hardware, and connecting them using zHyperLink cables. This creates a point-to-point connection between the CPU and I/O device, which reduces the I/O response time by up to 10 times, compared to IBM z High-Performance FICON® (zHPF). Such low response time is achieved by using synchronous I/O requests.

The advantages of synchronous I/O over asynchronous I/O

The IBM MQ logger task consists of a loop waiting for the next piece of data that needs to be written to the log. When that data is available the logger schedules the write, waits for it to complete, and then moves on to the next piece of data.

Traditional I/O is slower than the CPU, so it is most efficient to perform the I/O asynchronously to free the CPU for other tasks. Therefore traditional asynchronous I/O requires the logger task to be suspended, until the write completes. When the write completes, the logger task must wait for a CPU to become available, adding a short re-dispatch delay, as well as delays caused by repopulating the CPU cache.

zHyperLink provides much faster I/O times that are closer to CPU speed. Therefore with zHyperLink, I/O can be performed synchronously, which means the logger task does not get suspended during the write operation, removing re-dispatch and cache-related delays.

While the write is happening, the logger task is still actively using the CPU, which increases CPU usage compared to traditional I/O.

If the queue manager attempts to use zHyperLink, and the zHyperLink write fails, for example because of configuration issues, then the queue manager transparently falls back to traditional I/O.

Minimum Hardware Requirements

- IBM z14 or later
- DS8880 or later

Software Requirements

- zHyperLink Express is supported on z/OS 2.3 or later.
- The z/OS image must run in an LPAR, not as a guest under IBM z/VM®.
- zHyperLink requires IBM z High-Performance FICON (zHPF) to be enabled.

Using zHyperLink with IBM MQ active logs

In order to use zHyperLink with the active logs of a queue manager, you need to:

- Configure IBM MQ to use zHyperLink, and
- Ensure the active logs are on zHyperLink capable volumes.

  See Getting Started with IBM zHyperLink for z/OS for more information.

You can configure IBM MQ to use zHyperLink by using one of the following methods:

- Specify ZHYLINK(YES) in the log parameters.
- Issue the command SET LOG ZHYLINK(YES).

Notes:

- zHyperLink requires that zHyperWrite is switched on. This means in order to use ZHYLINK, ZHYWRITE must also be switched on in the log parameters. When only specifying ZHYLINK(YES) when ZHYWRITE(NO) is set on the queue manager, the ZHYWRITE parameter automatically overrides to YES.
- Explicitly trying to set ZHYLINK(YES) with ZHYWRITE(NO) results in an abnormal completion of the SET LOG command.
- Setting ZHYLINK=YES in the ZPRMs overrides ZHYWRITE to YES.

If you are experiencing any problems, see Troubleshooting zHyperLink for more information.

IBM MQ does not require that all active log data sets are on zHyperLink capable volumes, but you are advised to do so. If IBM MQ detects that some active log data sets are on zHyperLink capable volumes, and others are not, it issues message CSQJ601E and carries on processing.

IBM MQ checks whether active log data sets are zHyperLink capable when the data sets are first opened. Log data sets are opened either at queue manager start up, or when dynamically adding using the DEFINE LOG command. If the log data sets are made zHyperLink capable while a queue manager has them open, the queue manager does not detect this until it has been restarted.

If ZHYLINK(YES) is specified, then IBM MQ always attempts to use zHyperLink when writing to the active logs, regardless of whether the logs are on zHyperLink capable volumes. There are no negative effects of attempting to use zHyperLink if the logs are not on zHyperLink capable volumes.

You can use the output of the DISPLAY LOG command to indicate the status of zHyperLink for the current active log data sets:

```
Copy %Full zHyperWrite    Encrypted          DSName
  1     81 YES            NO                 MQTST.SUBSYS.MQDL.LOGCOPY1.DS001
  2     81 YES            NO                 MQTST.SUBSYS.MQDL.LOGCOPY2.DS001
Copy zHyperLink
  1  YES
  2  YES
```

The zHyperLink status is one of the following:

YES
: zHyperLink is enabled on the queue manager and will be attempted on all writes.

NO
: zHyperLink is not enabled on the queue manager and the dataset **is not** on zHyperLink capable volumes.

CAPABLE
zHyperLink is not enabled on the queue manager and the dataset **is** on a zHyperLink capable volume.

There are multiple additional SMF statistics for monitoring and understanding zHyperLink performance; see zHyperLink statistics for details.

Write sessions

When using zHyperLink, one or more zHyperLink write sessions are established with the DASD. Current DASD support a maximum of 64 concurrent write sessions so you should carefully consider which queue managers you enable zHyperLink on, and whether other subsystems, such as Db2 are also using zHyperLink for writing to the same DASD. If you run out of available write sessions then the queue manager automatically switches back to using traditional asynchronous I/O.

You can calculate the number of zHyperLink write sessions as follows:

- Number of log copies (either 1 or 2) * number of stripes per log copy * 2 if MetroMirror (PPRC) is used.

Therefore a queue manager in single logging mode with one stripe and no MetroMirror uses a single write session. A queue manager in dual logging mode, with two stripes and MetroMirror uses 8 write sessions.

Note: While MetroMirror results in twice as many write sessions being used, those write sessions are split evenly between the two mirrored DASD.

z/OS *Planning your log archive storage*

Use this topic to understand the different ways of maintaining your archive log data sets.

You can place archive log data sets on standard-label tapes, or DASD, and you can manage them by data facility hierarchical storage manager (DFHSM). Each z/OS logical record in an archive log data set is a VSAM control interval from the active log data set. The block size is a multiple of 4 KB.

Archive log data sets are dynamically allocated, with names chosen by IBM MQ. The data set name prefix, block size, unit name, and DASD sizes needed for such allocations are specified in the system parameter module. You can also choose, at installation time, to have IBM MQ add a date and time to the archive log data set name.

It is not possible to specify with IBM MQ, specific volumes for new archive logs, but you can use Storage Management routines to manage this. If allocation errors occur, offloading is postponed until the next time offloading is triggered.

If you specify dual archive logs at installation time, each log control interval retrieved from the active log is written to two archive log data sets. The log records that are contained in the pair of archive log data sets are identical, but the end-of-volume points are not synchronized for multivolume data sets.

Should your archive logs reside on tape or DASD?

When deciding whether to use tape or DASD for your archive logs, there are a number of factors that you should consider:

- Review your operating procedures before deciding about tape or disk. For example, if you choose to archive to tape, there must be enough tape drive when they are required. After a disaster, all subsystems might want tape drives and you might not have as many free tape drives as you expect.
- During recovery, archive logs on tape are available as soon as the tape is mounted. If DASD archives have been used, and the data sets migrated to tape using hierarchical storage manager (HSM), there is a delay while HSM recalls each data set to disk. You can recall the data sets before the archive log is used. However, it is not always possible to predict the correct order in which they are required.
- When using archive logs on DASD, if many logs are required (which might be the case when recovering a page set after restoring from a backup) you might require a significant quantity of DASD to hold all the archive logs.

- In a low-usage system or test system, it might be more convenient to have archive logs on DASD to eliminate the need for tape mounts.
- Both issuing a RECOVER CFSTRUCT command, and backing out a persistent unit of work, result in the log being read backwards. Tape drives with hardware compression perform badly on operations that read backwards. Plan sufficient log data on DASD to avoid reading backwards from tape.

Archiving to DASD offers faster recoverability but is more expensive than archiving to tape. If you use dual logging, you can specify that the primary copy of the archive log go to DASD and the secondary copy go to tape. This increases recovery speed without using as much DASD, and you can use the tape as a backup.

See "Changing the storage medium for archive logs" on page 164 for details of how you archive your logs from tape to DASD, and how you carry out the reverse process.

Archiving to tape

If you choose to archive to a tape device, IBM MQ can extend to a maximum of 20 volumes.

If you are considering changing the size of the active log data set so that the set fits on one tape volume, note that a copy of the BSDS is placed on the same tape volume as the copy of the active log data set. Adjust the size of the active log data set downward to offset the space required for the BSDS on the tape volume.

If you use dual archive logs on tape, it is typical for one copy to be held locally, and the other copy to be held off-site for use in disaster recovery.

Archiving to DASD volumes

IBM MQ requires that you catalog all archive log data sets allocated on non-tape devices (DASD). If you choose to archive to DASD, the CATALOG parameter of the CSQ6ARVP macro must be YES. If this parameter is NO, and you decide to place archive log data sets on DASD, you receive message CSQJ072E each time an archive log data set is allocated, although IBM MQ still catalogs the data set.

If the archive log data set is held on DASD, the archive log data sets can extend to another volume; multivolume is supported.

If you choose to use DASD, make sure that the primary space allocation (both quantity and block size) is large enough to contain either the data coming from the active log data set, or that from the corresponding BSDS, whichever is the larger of the two.

This minimizes the possibility of unwanted z/OS X'B37' or X'E37' abend codes during the offload process. The primary space allocation is set with the PRIQTY (primary quantity) parameter of the CSQ6ARVP macro.

Archive log data sets can exist on large or extended-format sequential data sets. SMS ACS routines now use DSNTYPE(LARGE) or DSNTYPE(EXT).

IBM MQ supports allocation of archive logs as extended format data sets. When extended format is used, the maximum archive log size is increased from 65535 tracks to the maximum active log size of 4GB. Archive logs are eligible for allocation in the extended addressing space (EAS) of extended address volumes (EAV).

Where the required hardware and software levels are available, allocating archive logs to a data class defined with COMPACTION using zEDC might reduce the disk storage required to hold archive logs. For more information, see IBM MQ for z/OS: Reducing storage occupancy with IBM zEnterprise Data Compression (zEDC) and zEnterprise Data Compression (zEDC) for more information.

The z/OS data set encryption feature can be applied to archive logs for queue managers running on IBM MQ. These archive logs must be allocated through Automatic Class Selection (ACS) routines to a data class defined with EXTENDED attributes, and a data set key label that ensures the data is AES encrypted.

Using SMS with archive log data sets

If you have MVS/DFP storage management subsystem (DFSMS) installed, you can write an Automatic Class Selection (ACS) user-exit filter for your archive log data sets, which helps you convert them for the SMS environment.

Such a filter, for example, can route your output to a DASD data set, which DFSMS can manage. You must exercise caution if you use an ACS filter in this manner. Because SMS requires DASD data sets to be cataloged, you must make sure the CATALOG DATA field of the CSQ6ARVP macro contains YES. If it does not, message CSQJ072E is returned; however, the data set is still cataloged by IBM MQ.

For more information about ACS filters, see Data sets that DFSMShsm dynamically allocates during aggregate backup processing.

z/OS *Changing the storage medium for archive logs*

The procedure for changing the storage medium used by archive logs.

About this task

This task describes how to change the storage medium used for archive logs, for example moving from archiving to tape to archiving to DASD.

You have a choice of how to make the changes:

1. Make the changes only using the CSQ6ARVP macro so that they are applied from the next time the queue manager restarts.
2. Make the changes using the CSQ6ARVP macro, and dynamically using the SET ARCHIVE command. This means that the changes apply from the next time the queue manager archives a log file, and persist after the queue manager restarts.

Procedure

1. Changing so archive logs are stored on DASD instead of tape:
   a) Read the section "Archiving to DASD volumes" on page 163 and review the CSQ6ARVP parameters.
   b) Make changes to the following parameters in CSQ6ARVP
      - Update the UNIT and, if necessary, the UNIT2 parameters.
      - Update the BLKSIZE parameter, as the optimal setting for DASD differs from tape.
      - Set the PRIQTY and SECQTY parameters to be large enough to hold the largest of the active log or BSDS.
      - Set the CATALOG parameter to be YES.
      - Confirm the ALCUNIT setting is what you want. You should use BLK, because it is independent of the device type.
      - Set the ARCWTOR parameter to NO if it is not already.
2. Changing so archive logs are stored on tape instead of DASD:
   a) Read the section "Archiving to tape" on page 163, and review the CSQ6ARVP parameters.
   b) Make changes to the following parameters in CSQ6ARVP:
      - Update the UNIT and, if necessary, the UNIT2 parameters.
      - Update the BLKSIZE parameter, as the optimal setting for tape differs from DASD.
      - Confirm the ALCUNIT setting is what you want. You should use BLK, because it is independent of the device type.
      - Review the setting of the ARCWTOR parameter.

z/OS *How long do I need to keep archive logs*

Use the information in this section to help you plan your backup strategy.

You specify how long archive logs are kept in days , using the ARCRETN parameter in USING CSQ6ARVP or the SET SYSTEM command. After this period the data sets can be deleted by z/OS.

You can manually delete archive log data sets when they are no longer needed.

- The queue manager might need the archive logs for recovery.

  The queue manager can only keep the most recent 1000 archives in the BSDS, When the archive logs are not in the BSDS they cannot be used for recovery, and are only of use for audit, analysis, or replay type purposes.
- You might want to keep the archive logs so that you can extract information from the logs. For example, extracting messages from the log, and reviewing which user ID put or got the message.

The BSDS contains information on logs and other recovery information. This data set is a fixed size. When the number of archive logs reaches the value of MAXARCH in CSQ6LOGP, or when the BSDS fills up, the oldest archive log information is overwritten.

There are utilities to remove archive log entries from the BSDS, but in general, the BSDS wraps and overlays the oldest archive log record.

When is an archive log needed

You need to back up your page sets regularly. The frequency of backups determines which archive logs are needed in the event of losing a page set.

You need to back up your CF structures regularly. The frequency of backups determines which archive logs are needed in the event of losing data in the CF structure.

The archive log might be needed for recovery. The following information explains when the archive log might be needed, where there are problems with different IBM MQ resources.

Loss of a page set

You must recover your system from your backup and restart the queue manager.

You need the logs from when the backup was taken, as well as up to three log data sets prior to the backup being taken.

All LPARs lose connectivity to a CF structure, or the structure is unavailable

Use the RECOVER CFSTRUCT command to recover the structure.

Structure recovery requires the logs from all queue managers that have accessed the structure since the last backup (back to the time when the backup was taken) plus the structure backup itself in the log of the queue manager that took the backup.

If you have been doing frequent backups of the CF structures, the data should be in active logs, and you should not need archive logs.

If there is no recent backup of the CF structure, you might need archive logs.

Note: All non persistent messages will be lost; all persistent messages will be re-created by performing the following tasks:

1. Reading the last CF structure backup from the log
2. Reading the logs from all queue managers that have used the structure
3. Merging updates since the backup

Administration structure rebuild

If you need to rebuild the administration structure, the information is read from the last checkpoint of the log for each queue manager in the QSG.

If a queue manager is not active, another queue manager in the QSG reads the log.

You should not need archive logs.

Loss of an SMDS data set

If you lose an SMDS data set, or the data set gets corrupted, the data set becomes unusable and the status for it is set to FAILED. The CF structure is unchanged.

In order to restore the SMDS data set, you need to:

1. Redefine the SMDS data set, and
2. Recover the CF structure by issuing the RECOVER CFSTRUCT command.

   Note: All non persistent messages on the CF structure will be lost; all persistent messages will be restored.

The requirement for queue manager logs is the same as for recovering from a structure that is unavailable.

z/OS *Planning to increase the maximum addressable log range*

You can increase the maximum addressable log range by configuring your queue manager to use a larger log relative byte address (RBA).

The log RBA size was increased from IBM MQ for z/OS 8.0. For an overview of this change, see Larger log Relative Byte Address.

Queue managers created at IBM MQ 9.3.0 or later, have 8 byte log RBA enabled by default and, therefore, do not require conversion.

You can convert your queue managers to use 8 byte log RBA values at any time. A queue sharing group can contain some queue managers with 8 byte log RBA enabled, and some queue managers with 6 byte log RBA enabled.

Undoing the change

The change cannot be backed out.

How long does it take?

The change requires a queue manager restart. Stop the queue manager, run the CSQJUCNV utility against the bootstrap data set (BSDS), or data sets, to create new data sets, rename these bootstrap data sets, and restart the queue manager. The CSQJUCNV utility usually takes a few seconds to run.

What impact does this have?

- With 8 byte log RBA in use, every write of data to the log data sets has additional bytes. Therefore, for a workload consisting of persistent messages there is a small increase in the amount of data written to the logs.
- Data written to a page set, or coupling facility (CF) structure, is not affected.

Related tasks

Implementing the larger log Relative Byte Address

z/OS Planning your channel initiator

The channel initiator provides communications between queue managers, and runs in its own address space.

There are two types of connections:

1. Application connections to a queue manager over a network. These are known as client channels.
2. Queue manager to queue manager connections. These are known as MCA channels.

Listeners

A channel listener program listens for incoming network requests and starts the appropriate channel when that channel is needed. To process inbound connections the channel initiator needs at least one IBM MQ listener task configured. A listener can either be a TCP listener, or a LU 6.2 listener.

Each listener requires a TCP port or LU name.

Note that you can have more than one listener for each channel initiator.

TCP/IP

A channel initiator can operate with more than one TCP stack on the same z/OS image. For example, one TCP stack could be for internal connections, and another TCP stack for external connections.

When you define an output channel:

1. You set the destination host and port of the connection. This can be either:
   - an IP address, for example `10.20.4.6`
   - a host name, for example `mvs-prod.myorg.com`

   If you use a host name to specify the destination, IBM MQ uses the Domain Name System (DNS) to resolve the IP address of the destination.
2. If you are using multiple TCP stacks you can specify the **LOCLADDR** parameter on the channel definition, which specifies the IP Stack address to be used.

You should plan to have a highly available DNS server, or DNS servers. If the DNS is not available, outbound channels might not be able to start, and channel authentication rules that map an incoming connection using a host name cannot be processed.

APPC and LU 6.2

If you are using APPC, the channel initiator needs an LU name, and configuration in APPC.

Queue sharing groups

To provide a single system image, and allow an incoming IBM MQ connection request to go to any queue manager in the queue sharing group, you need to do some configuration. For example:

1. A hardware network router. This router has one IP address seen by the enterprise, and can route the initial request to any queue manager connected to this hardware.
2. A Virtual IP address (VIPA). An enterprise wide IP address is specified, and that address can be routed to any one of the TCP stacks in a sysplex. The TCP stack can then route it to any listening queue manager in the sysplex.

Protecting IBM MQ traffic

You can configure IBM MQ to use TLS connections to protect data on the wire. To use TLS you need to use digital certificates and key rings.

You also need to work with the personnel at the remote end of the channel, to ensure that you have compatible IBM MQ definitions and compatible certificates.

You can control which connections can connect to IBM MQ and the user ID, based on

- IP address
- Client user ID
- Remote queue manager, or
- Digital certificate (see Channel Authentication Records)

It is also possible to restrict client applications by ensuring that they supply a valid user ID and password (see Connection Authentication).

You can get the channel initiator working, and then configure each channel to use TLS, one at a time.

Monitoring the channel initiator

There are MQSC commands that give information about the channel initiator and channels:

- The DISPLAY CHINIT command gives information about the channel initiator, and active listeners.
- The DISPLAY CHSTATUS command displays the activity and status of a channel.

The channel initiator can also produce SMF records with information about the channel initiator tasks and channel activity. See “Planning for channel initiator SMF data” on page 169 for more information.

The channel initiator emits messages to the job log when channels start and stop. Automation in your enterprise can use these messages to capture status. As some channels are active for only a few seconds, many messages can be produced. You can suppress these messages either by using the z/OS message processing facility, or by setting **EXCLMSG** with the SET SYSTEM command.

Configuring your IBM MQ channel definitions

When you have many queue managers connected together it can be hard to manage all the object definitions. Using IBM MQ clustering can simplify this.

You specify two queue managers as full repositories. Other queue managers need one connection to, and one connection from, one of the repositories. When connections to other queue managers are needed, the queue manager creates and starts channels automatically.

If you are planning to have a large number of queue managers in a cluster, you should plan to have queue managers that act as dedicated repositories and have no application traffic.

See “Planning your distributed queues and clusters” on page 19 for more information.

Actions before you configure the channel initiator

1. Decide if you are using TCP/IP or APPC.
2. If you are using TCP, allocate at least one port for IBM MQ.
3. If you need a a DNS server, configure the server to be highly available if required.
4. If you are using APPC, allocate an LU name, and configure APPC.

Actions after you have configured the channel initiator, before you go into production

1. Plan what connections you will have:
   a. Client connections from remote applications.
   b. MCA channels to and from other queue managers. Typically you have a channel to and from each remote queue manager.
2. Set up clustering, or join an existing clustering environment.
3. Consider whether you need to use multiple TCP stacks, VIPA, or an external router for availability in front of the channel initiator.
4. If you are planning on using TLS:
   a. Set up the key ring
   b. Set up certificates
5. If you are planning on using channel authentication:
   a. Decide the criteria for mapping inbound sessions to MCA user IDs

b. Enable reverse DNS lookup by setting the queue manager parameter **REVDNS**

c. Review security. For example, delete the default channels, and specify user IDs with only the necessary authority in the **MCAUSER** attribute for a channel.

6. Capture the accounting and statistics SMF records produced by the channel initiator and post process them.
7. Automate the monitoring of job log messages.
8. If necessary, tune your network environment to improve throughput. With TCP, large send and receive buffers improve throughput. You can force MQ to use specific TCP buffer sizes using the commands:

```
RECOVER QMGR(TUNE CHINTCPRBDYNSZ nnnnn)
RECOVER QMGR(TUNE CHINTCPSBDYNSZ nnnnn)
```

which sets the SO_RCVBUF, and SO_SNDBUF, for the channels to the size in bytes specified in nnnnn.

Related concepts

“Planning for your queue manager” on page 138

When you are setting up a queue manager, your planning should allow for the queue manager to grow, so that the queue manager meets the needs of your enterprise.

z/OS Planning for channel initiator SMF data

You need to plan the implementation of collecting SMF data for the channel initiator.

The channel initiator produces two types of record:

- Statistics data with information about the channel initiator and the tasks within it.
- Channel accounting data with information similar to the DISPLAY CHSTATUS command.

You start collecting statistics data using the command:

```
START TRACE(STAT) CLASS(4)
```

and stop it using the command:

```
STOP TRACE(STAT) CLASS(4)
```

You start collecting accounting data using the command:

```
START TRACE(ACCTG) CLASS(4)
```

and stop it using the command:

```
STOP TRACE(ACCTG) CLASS(4)
```

You can control which channels have accounting data collected for using the **STATCHL** attribute on the channel definition or the queue manager.

- For client channels, you must set **STATCHL** at the queue manager level.
- For automatically defined cluster sender channels, you can control the collection of accounting data with the **STATACLS** queue manager attribute.

The default value of **STATCHL** for the queue manager is OFF. In order to collect channel accounting data you must change the value of **STATCHL** from the default on either the queue manager or channel definition, in addition to starting class 4 accounting trace.

The SMF records are produced when:

- From IBM MQ for z/OS 9.3.0 onwards, the time interval indicated by the CSQ6SYSP **STATIME** or **ACCTIME** parameters has elapsed; or, if **STATIME** or **ACCTIME** is zero on the SMF data collection broadcast. The requests to collect SMF data for the channel initiator and the queue manager are synchronized.
- A `STOP TRACE(ACCTG) CLASS(4)` or `STOP TRACE(STAT) CLASS(4)` command is issued, or

- The channel initiator is shut down. At this point any SMF data is written out.

If a channel stops during the SMF interval, accounting data is written to SMF the next time the SMF processing runs. If a client connects, does some work and disconnects, then reconnects and disconnects, there are two sets of channel accounting data produced.

The statistics data normally fits into one SMF record, however, multiple SMF records might be created if a large number of tasks are in use.

Accounting data is gathered for each channel for which it is enabled, and normally fits into one SMF record. However, multiple SMF records might be created if a large number of channels are active.

The cost of collecting the channel initiator SMF data is small. Typically the increase in CPU usage is under a few percent, and often within measurement error.

Before you use this function you need to work with your z/OS systems programmer to ensure that SMF has the capacity for the additional records, and that they change their processes for extracting SMF records to include the new SMF data.

For channel initiator statistics data, the SMF record type is 115 and sub-type 231.

For channel initiator accounting data, the SMF record type is 116 and sub-type 10.

You can write your own programs to process this data, or use the SupportPac MP1B that contains a program, MQSMF, for printing the data, and creating data in Comma Separated Values (CSV) format suitable for importing into a spread sheet.

If you are experiencing issues with capturing channel initiator SMF data, see Dealing with issues when capturing SMF data for the channel initiator (CHINIT) for further information.

Related tasks

Interpreting IBM MQ performance statistics

Troubleshooting channel accounting data

z/OS Planning your z/OS TCP/IP environment

To get the best throughput through your network, you must use TCP/IP send and receive buffers with a size of 64 KB, or greater. With this size, the system optimizes its buffer sizes.

See What is Dynamic Right Sizing for High Latency Networks? for more information.

You can check your system buffer size by using the following Netstat command, for example:

```
TSO NETSTAT ALL (CLIENT csq1CHIN
```

The results display much information, including the following two values:

```
ReceiveBufferSize: 0000065536
SendBufferSize: 0000065536
```

65536 is 64 KB. If your buffer sizes are less than 65536, you must work with your network team to increase the **TCPSENDBFRSIZE** and **TCPRCVBUFRSIZE** values in the `PROFILE DDName` in the TCPIP procedure. For example, you might use the following command:

```
TCPCONFIG TCPSENDBFRSZE 65536 TCPRCVBUFRSIZE 65536
```

If you are unable to change your system-wide **TCPSENDBFRSIZE** or **TCPRCVBUFRSIZE** settings, contact IBM Support.

z/OS Planning your queue sharing group (QSG)

The easiest way to implement a shared queuing environment, is to configure a queue manager, add that queue manager to a QSG, then add other queue managers to the QSG.

A queue sharing group uses Db2 tables to store configuration information. There is one set of tables used by all QSGs that share the same Db2 data sharing group.

Shared queue messages are stored in a structure in a coupling facility (CF). Each QSG has its own set of CF structures. You need to configure the structures to meet your needs.

Messages over 63KB in size cannot be stored in the CF. You need to use either Shared Message Data Sets (SMDS) or Db2 for these messages.

Message profiles and capacity planning

You should understand the message profile of your shared queue messages. The following are examples of factors that you need to consider:

- Average, and maximum message size
- The typical queue depth, and exception queue depth. For example, you might need to have enough capacity to hold messages for a whole day, and the typical queue depth is under 100 messages.

If the message profile changes, you can increase the size of the structures, or implement SMDS, at a later date.

If you want to be able to handle a large peak volume of messages, you can configure IBM MQ to offload messages to SMDS when the usage of the structure reaches user specified thresholds.

You need to decide if you want to duplex the CF structures. This is controlled by the CF structure definition in the CFRM policy:

1. A duplexed structure uses two coupling facilities. If there is a problem with one CF, there is no interruption to the service, and the structure can be rebuilt on a third CF, if one is available. Duplexed structures can significantly impact the performance of operations on shared queues.
2. If the structure is not duplexed, then a problem with the CF means that shared queues on structures in that CF will become unavailable until the structure can be rebuilt in another CF.

   IBM MQ can be configured to automatically rebuild structures in another CF in this case. Persistent messages will be recovered from the logs of the queue managers.

Note that it is easy to change the CF definitions.

You can define a structure so that it can hold nonpersistent messages only, or so that it can hold persistent and nonpersistent messages.

Structures that can hold persistent messages need to be backed up periodically. Back up your CF structures at least every hour to minimize the time needed to recover the structure in the event of a failure. The backup is stored in the log data set of the queue manager performing the backup.

If you are expecting to have a high throughput of messages on your shared queues, it is best practice to have a dedicated queue manager for backing up the CF structures. This reduces the time needed to recover the structures, as a less data needs to be read from queue manager logs.

Channels

To provide a single system image for applications connecting into an IBM MQ QSG, you can define shared input channels. If these are set up, then a connection coming into the queue sharing group environment, can go to any queue manager in the QSG.

You might need to set up a network router, or Virtual IP address (VIPA) for these channels.

You can define shared output channels. A shared output channel instance can be started from any queue manager in the QSG.

See Shared channels for more information.

Security

You protect IBM MQ resources using an external security manager. If you are using RACF®, the RACF profiles are prefixed with the queue manager name. For example, a queue named APPLICATION.INPUT would be protected using a profile in the MQQUEUE class named `qmgrName.APPLICATION.INPUT`.

When using a queue sharing group you can continue to protect resources with profiles prefixed with the queue manager name, or you can prefix profiles with the queue sharing group name. For example `qsgName.APPLICATION.INPUT`.

You should aim to use profiles prefix with the queue sharing group name because this means there is a single definition for all queue managers, saving you work, and preventing a mismatch in definitions between queue managers.

Related concepts

"Planning for your queue manager" on page 138
When you are setting up a queue manager, your planning should allow for the queue manager to grow, so that the queue manager meets the needs of your enterprise.

z/OS Planning your coupling facility and offload storage environment

Use this topic when planning the initial sizes, and formats of your coupling facility (CF) structures, and shared message data set (SMDS) environment or Db2 environment.

This section contains information about the following topics:

- "Defining coupling facility resources" on page 172
  - Deciding your offload storage mechanism
  - Planning your structures
  - Planning the size of your structures
  - Mapping shared queues to structures
- "Planning your shared message data set (SMDS) environment" on page 178
- "Planning your Db2 environment" on page 181

Defining coupling facility resources

If you intend to use shared queues, you must define the coupling facility structures that IBM MQ will use in your CFRM policy. To do this you must first update your CFRM policy with information about the structures, and then activate the policy.

Your installation probably has an existing CFRM policy that describes the coupling facilities available. The Administrative data utility is used to modify the contents of the policy based on textual statements you provide. You must add statements to the policy that defines the names of the new structures, the coupling facilities that they are defined in, and what size the structures are.

The CFRM policy also determines whether IBM MQ structures are duplexed and how they are reallocated in failure scenarios. Shared queue recovery contains recommendations for configuring CFRM for resilience to failures that affect the coupling facility.

Deciding your offload storage environment

The message data for shared queues can be offloaded from the coupling facility and stored in either a Db2 table or in an IBM MQ managed data set called a *shared message data set* (SMDS). Messages which are

too large to store in the coupling facility (that is, larger than 63 KB) must always be offloaded, and smaller messages can optionally be offloaded to reduce coupling facility space usage.

For more information, see Specifying offload options for shared messages.

Planning your structures

A queue sharing group (QSG) requires a minimum of two structures to be defined. The first structure, known as the administrative structure, is used to coordinate IBM MQ internal activity across the queue sharing group. No user data is held in this structure. It has a fixed name of *qsg-name*CSQ_ADMIN (where *qsg-name* is the name of your queue sharing group). Subsequent structures are known as application structures, and are used to hold the messages on IBM MQ shared queues. Each structure can hold up to 512 shared queues.

An application structure named *qsg-name*CSQSYSAPPL is used for system queues. Defining this structure is optional, but it is required in order to use certain features. By default, the SYSTEM.QSG.CHANNEL.SYNCQ and SYSTEM.QSG.UR.RESOLUTION.QUEUE queues are defined on the *qsg-name*CSQSYAPPL structure.

Using multiple structures

A queue sharing group can connect to up to 64 coupling facility structures. One of these structures must be the administration structure. If it is defined, another of these structures might be the *qsg-name*CSQSYSAPPL structure. You can use up to 63 (62 if *qsg-name*CSQSYSAPPL is defined) structures for message data. You might choose to use multiple application structures for any of the following reasons:

- You have some queues that are likely to hold a large number of messages and so require all the resources of an entire coupling facility.
- You have a requirement for a large number of shared queues, so they must be split across multiple structures because each structure can contain only 512 queues.
- RMF reports on the usage characteristic of a structure suggest that you should distribute the queues it contains across a number of coupling facilities.
- You want some queue data to held in a physically different coupling facility from other queue data for data isolation reasons.
- Recovery of persistent shared messages is performed using structure level attributes and commands, for example BACKUP CFSTRUCT. To simplify backup and recovery, you could assign queues that hold nonpersistent messages to different structures from those structures that hold persistent messages.

When choosing which coupling facilities to allocate the structures in, consider the following points:

- Your data isolation requirements.
- The volatility of the coupling facility (that is, its ability to preserve data through a power outage).
- Failure independence between the accessing systems and the coupling facility, or between coupling facilities.
- The level of coupling facility control code (CFCC) installed on the coupling facility (IBM MQ requires Level 9 or higher).

Planning the size of your structures

The administrative structure

The administrative structure (*qsg-name*CSQ_ADMIN) must be large enough to contain 1000 list entries for each queue manager in the queue sharing group. When a queue manager starts, the structure is checked to see if it is large enough for the number of queue managers currently *defined*

to the queue sharing group. Queue managers are considered as being defined to the queue sharing group if they have been added by the CSQ5PQSG utility. You can check which queue managers are defined to the group with the MQSC DISPLAY GROUP command.

Note: When calculating the size of the structure, you should allow for the size of large units of work, in addition to the number of queue managers in the queue sharing group.

Table 22 on page 174 shows the minimum required size for the administrative structure for various numbers of queue managers defined in the queue sharing group. These sizes were established for a CFCC level 14 coupling facility structure; for higher levels of CFCC, they probably need to be larger.

Table 22. Minimum administrative structure sizes

Number of queue managers defined in queue sharing group	Required storage
1	6144 KB
2	6912 KB
3	7976 KB
4	8704 KB
5	9728 KB
6	10496 KB
7	11520 KB
8	12288 KB
9	13056 KB
10	14080 KB
11	14848 KB
12	15616 KB
13	16640 KB
14	17408 KB
15	18176 KB
16	19200 KB
17	19968 KB
18	20736 KB
19	21760 KB
20	22528 KB
21	23296 KB
22	24320 KB
23	25088 KB
24	25856 KB
25	27136 KB
26	27904 KB
27	28672 KB
28	29696 KB

Table 22. Minimum administrative structure sizes (continued)	
Number of queue managers defined in queue sharing group	**Required storage**
29	30464 KB
30	31232 KB
31	32256 KB

When you add a queue manager to an existing queue sharing group, the storage requirement might have increased beyond the size recommended in Table 22 on page 174. If so, use the following procedure to estimate the required storage for the *qsg-name*CSQ_ADMIN structure:

1. Issue MQSC command **DISPLAY CFSTATUS(CSQ_ADMIN)** on an existing member of the queue sharing group.
2. Extract the ENTSMAX information for the CSQ_ADMIN structure.
3. If this number is less than 1000 times the total number of queue managers you want to define in the queue sharing group, increase the structure size.

Application structures

The size of the application structures required to hold IBM MQ messages depends on the likely number and size of the messages to be held on a structure concurrently.

The graph in Figure 42 on page 176 shows how large you should make your CF structures to hold the messages on your shared queues. To calculate the allocation size you need the following information:

- The average size of messages on your queues.
- The total number of messages likely to be stored in the structure.

Find the number of messages along the horizontal axis. Select the curve that corresponds to your message size and determine the required value from the vertical axis. For example, for 200 000 messages of length 1 KB gives a value in the range 256 through 512 MB.

Table 23 on page 176 provides the same information in tabular form.

Figure 42. Calculating the size of a coupling facility structure

Use this table to help calculate how large to make your coupling facility structures:

Table 23. Calculating the size of a coupling facility structure

Number of messages	1 KB	2 KB	4 KB	8 KB	16 KB	32 KB	63 KB
100	6 MB	6 MB	7 MB	7 MB	8 MB	10 MB	14 MB
1000	8 MB	9 MB	12 MB	17 MB	27 MB	48 MB	88 MB
10000	25 MB	38 MB	64 MB	115 MB	218 MB	423 MB	821 MB
100000	199 MB	327 MB	584 MB	1097 MB	2124 MB	4177 MB	8156 MB

Your CFRM policy should include the following statements:

- `INITSIZE` is the size in KB that the structure is allocated with when the first queue manager connects to it.
- `SIZE` is the maximum size that the structure can attain.
- `FULLTHRESHOLD` sets the percentage value of the threshold at which z/OS issues message IXC585E to indicate that the structure is getting full.

A best practice is to ensure that INITSIZE and SIZE are within a factor of 2. For example, with the figures determined previously, you might include the following statements:

```
STRUCTURE NAME(structure-name)
INITSIZE(value from graph in KB, that is, multiplied by 1024)
SIZE(something larger)
FULLTHRESHOLD(85)
```

```
STRUCTURE NAME(QSG1APPLICATION1)
INITSIZE(262144) /* 256 MB */
SIZE(524288)     /* 512 MB */
FULLTHRESHOLD(85)
```

If the structure use reaches the threshold where warning messages are issued, intervention is required. You might use IBM MQ to inhibit MQPUT operations to some of the queues in the structure to prevent applications from writing more messages, start more applications to get messages from the queues, or quiesce some of the applications that are putting messages to the queue.

Alternatively, you can use z/OS facilities to alter the structure size in place. The following z/OS command:

```
SETXCF START,ALTER,STRNAME=structure-name,SIZE=newsize
```

alters the size of the structure to *newsize*, where *newsize* is a value that is less than the value of SIZE specified on the CFRM policy for the structure, but greater than the current coupling facility size.

You can monitor the use of a coupling facility structure with the MQSC DISPLAY CFSTATUS command.

If no action is taken and a queue structure fills up, an MQRC_STORAGE_MEDIUM_FULL return code is returned to the application. If the administration structure becomes full, the exact symptoms depend on which processes experience the error, but they might include the following problems:

- No responses to commands.
- Queue manager failure as a result of problems during commit processing.

The CSQSYSAPPL structure

The *qsg-name*CSQSYSAPPL structure is an application structure for system queues. Table 3 demonstrates an example of how to estimate the message data sizes for the default queues defined on the *qsg-name*CSQSYSAPPL structure.

Table 24. Table showing CSQSYSAPPL usage against sizing.

***qsg-name*CSQSYSAPPL usage**	**Sizing**
SYSTEM.QSG.CHANNEL.SYNCQ	2 messages of 500 bytes per active instance of a shared channel
SYSTEM.QSG.UR.RESOLUTION.QUEUE	1000 messages of 2 KB

The suggested initial structure definition values are as follows:

```
STRUCTURE NAME(qsg-nameCSQSYSAPPL)
INITSIZE(20480)        /* 20 MB */
SIZE(30720)            /* 30 MB */
FULLTHRESHOLD(85)
```

These values can be adjusted depending on your use of shared channels and group units of recovery.

Mapping shared queues to structures

To define an application structure to IBM MQ, use the DEFINE CFSTRUCT command. When you define a structure to IBM MQ, do not include the QSG name prefix in the structure name. For example, to define an

application structure to IBM MQ that has the name *qsg-name*APPLICATION1 in the CFRM policy, issue the following command:

```
DEFINE CFSTRUCT(APPLICATION1)
```

The CFSTRUCT attribute of the queue definition is used to map the queue to a structure. Specify the name of the CF structure without the QSG name prefix in this attribute. For example, the following command defines a shared queue on the APPLICATION1 structure:

```
DEFINE QLOCAL(myqueue) QSGDISP(SHARED) CFSTRUCT(APPLICATION1)
```

z/OS *Planning your shared message data set (SMDS) environment*

If you are using queue sharing groups with SMDS offloading, IBM MQ needs to connect to a group of shared message data sets. Use this topic to help understand the data set requirements, and configuration required to store IBM MQ message data.

A *shared message data set* (described by the keyword SMDS) is a data set used by a queue manager to store offloaded message data for shared messages stored in a coupling facility structure.

Note: When defining SMDS data sets for a structure, you must have one for each queue manager.

When this form of data offloading is enabled, the **CFSTRUCT** requires an associated group of shared message data sets, one data set for each queue manager in the queue sharing group. The group of shared message data sets is defined to IBM MQ using the **DSGROUP** parameter on the **CFSTRUCT** definition. Additional parameters can be used to supply further optional information, such as the number of buffers to use and expansion attributes for the data sets.

Each queue manager can write to the data set which it owns, to store shared message data for messages written through that queue manager, and can read all of the data sets in the group.

A list describing the status and attributes for each data set associated with the structure is maintained internally as part of the **CFSTRUCT** definition, so each queue manager can check the definition to find out which data sets are currently available.

This data set information can be displayed using the **DISPLAY CFSTATUS TYPE(SMDS)** command to display current status and availability, and the **DISPLAY SMDS** command to display the parameter settings for the data sets associated with a specified **CFSTRUCT**.

Individual shared message data sets are effectively identified by the combination of the owning queue manager name (usually specified using the **SMDS** keyword) and the **CFSTRUCT** structure name.

This section describes the following topics:

- The DSGROUP parameter
- The DSBLOCK parameter
- Shared message data set characteristics
- Shared message data set space management
- Access to shared message data sets
- Creating a shared message data set
- Shared message data set performance and capacity considerations
- Activating a shared message data set

See DEFINE CFSTRUCT for details of these parameters.

For information on managing your shared message data sets, see Managing shared message data sets for further details.

The DSGROUP parameter

The **DSGROUP** parameter on the **CFSTRUCT** definition identifies the group of data sets in which large messages for that structure are to be stored. Additional parameters may be used to specify the logical

block size to be used for space allocation purposes and values for the buffer pool size and automatic data set expansion options.

The **DSGROUP** parameter must be set up before offloading to data sets can be enabled.

- If a new **CFSTRUCT** is being defined at **CFLEVEL(5)** and the option **OFFLOAD(SMDS)** is specified or assumed, then the **DSGROUP** parameter must be specified on the same command.
- If an existing **CFSTRUCT** is being altered to increase the **CFLEVEL** to **CFLEVEL(5)** and the option **OFFLOAD(SMDS)** is specified or assumed, then the **DSGROUP** parameter must be specified on the same command if it is not already set.

The DSBLOCK parameter

Space within each data set is allocated to queues as logical blocks of a fixed size (usually 256 KB) specified using the **DSBLOCK** parameter on the **CFSTRUCT** definition, then allocated to individual messages as ranges of pages of 4 KB (corresponding to the physical block size and control interval size) within each logical block. The logical block size also determines the maximum amount of message data that can be read or written in a single I/O operation, which is the same as the buffer size for the SMDS buffer pool.

A larger value of the **DSBLOCK** parameter can improve performance for very large messages by reducing the number of separate I/O operations. However, a smaller value decreases the amount of buffer storage required for each active request. The default value for the **DSBLOCK** parameter is 256 KB, which provides a reasonable balance between these requirements, so specifying this parameter might not normally be necessary.

Shared message data set characteristics

A shared message data set is defined as a VSAM linear data set (LDS). Each offloaded message is stored in one or more blocks in the data set. The stored data is addressed directly by information in the coupling facility entries, like an extended form of virtual storage. There is no separate index or similar control information stored in the data set itself.

The direct addressing scheme means that for messages which fit into one block, only a single I/O operation is needed to read or write the block. When a message spans more than one block, the I/O operations for each block can be fully overlapped to minimize elapsed time, provided that sufficient buffers are available.

The shared message data set also contains a small amount of general control information, consisting of a header in the first page, which includes recovery and restart status information, and a space map checkpoint area which is used to save the free block space map at queue manager normal termination.

Shared message data set space management

As background information for capacity, performance and operational considerations, it might be useful to understand the concepts of how space in shared message data sets is managed by the queue managers.

Free space in each shared message data set is tracked by its owning queue manager using a space map which indicates the number of pages in use within each logical block. The space map is maintained in main storage while the data set is open and saved in the data set when it is closed normally. (In recovery situations the space map is automatically rebuilt by scanning the messages in the coupling facility structure to find out which data set pages are currently in use).

When a shared message with offloaded message data is being written, the queue manager allocates a range of pages for each message block. If there is a partly used current logical block for the specified queue, the queue manager allocates space starting at the next free page in that block, otherwise it allocates a new logical block. If the whole message does not fit within the current logical block, the queue manager splits the message data at the end of the logical block and allocates a new logical block for the next message block. This is repeated until space has been allocated for the whole message. Any unused space in the last logical block is saved as the new current logical block for the queue. When the data set

is closed normally, any unused pages in current logical blocks are returned to the space map before it is saved.

When a shared message with offloaded message data has been read and is ready to be deleted, the queue manager processes the delete request by transferring the coupling facility entry for the message to a clean-up list monitored by the owning queue manager (which may be the same queue manager). When entries arrive on this list, the owning queue manager reads and deletes the entries and returns the freed ranges of pages to the space map. When all used pages in a logical block have been freed the block becomes available for reuse.

Access to shared message data sets

Each shared message data set must be on shared direct access storage which is accessible to all queue managers in the queue sharing group.

During normal running, each queue manager opens its own shared message data set for read/write access, and opens any active shared message data sets for other queue managers for read-only access, so it can read messages stored by those queue managers. This means that each queue manager userid requires at least UPDATE access to its own shared message data set and READ access to all other shared message data sets for the structure.

If it is necessary to recover shared message data sets using **RECOVER CFSTRUCT**, the recovery process can be executed from any queue manager in the queue sharing group. A queue manager which may be used to perform recovery processing requires UPDATE access to all data sets that it may need to recover

Creating a shared message data set

Each shared message data set should normally be created before the corresponding **CFSTRUCT** definition is created or altered to enable the use of this form of message offloading, as the **CFSTRUCT** definition changes will normally take effect immediately, and the data set will be required as soon as a queue manager attempts to access a shared queue which has been assigned to that structure. A sample job to allocate and pre-format a shared message data set is provided in SCSQPROC(CSQ4SMDS). The job must be customized and run to allocate a shared message data set for each queue manager which uses a CFSTRUCT with OFFLOAD(SMDS).

If the queue manager finds that offload support has been enabled and tries to open its shared message data set but it has not yet been created, the shared message data set will be flagged as unavailable. The queue manager will then be unable to store any large messages until the data set has been created and the queue manager has been notified to try again, for example using the **START SMDSCONN** command.

A shared message data set is created as a VSAM linear data set using an Access Method Services **DEFINE CLUSTER** command. The definition must specify **SHAREOPTIONS(2 3)** to allow one queue manager to open it for write access and any number of queue managers to read it at the same time. The default control interval size of 4 KB must be used. If the data set may need to expand beyond 4 GB, it must be defined using an SMS data class which has the VSAM extended addressability attribute. A shared message data set is eligible to reside in the extended addressing space (EAS) part of an extended address volumes (EAV).

Each shared message data set can either be empty or pre-formatted to binary zeros (using **CSQJUFMT** or a similar utility such as the sample job SCSQPROC(CSQ4SMDS)), before its initial use. If it is empty or only partly formatted when it is opened, the queue manager automatically formats the remaining space to binary zeros.

Shared message data set performance and capacity considerations

Each shared message data set is used to store offloaded data for shared messages written to the associated **CFSTRUCT** by the owning queue manager, from regions within the same system. Each message that is offloaded takes up to 768 bytes of CF storage, made up of 256 bytes for the entry and 512 bytes for the two elements of header and descriptor. Each offloaded message is stored in one or more pages (physical blocks of size 4 KB) in the data set.

The data set space required for a given number of offloaded messages can therefore be estimated by rounding up the overall message size (including the descriptor) to the next multiple of 4 KB and then multiplying by the number of messages.

As for a page set, when a shared message data set is almost full, it can optionally be expanded automatically. The default behavior for this automatic expansion can be set using the **DSEXPAND** parameter on the **CFSTRUCT** definition. This setting can be overridden for each queue manager using the **DSEXPAND** parameter on the **ALTER SMDS** command. Automatic expansion is triggered when the data set reaches 90% full and more space is required. If expansion is allowed but an expansion attempt is rejected by VSAM because no secondary space allocation was specified when the data set was defined, expansion is retried using a secondary allocation of 20% of the current size of the data set.

Provided that the shared message data set is defined with the extended addressability attribute, the maximum size is only limited by VSAM considerations to a maximum of 16 TB or 59 volumes. This is significantly larger than the 64 GB maximum size of a local page set.

Activating a shared message data set

When a queue manager has successfully connected to an application coupling facility structure, it checks whether that structure definition specifies offloading using an associated **DSGROUP** parameter. If so, the queue manager allocates and opens its own shared message data set for write access, then it opens for read access any existing shared message data sets owned by other queue managers.

When a shared message data set is opened for the first time (before it has been recorded as active within the queue sharing group), the first page will not yet contain a valid header. The queue manager fills in header information to identify the queue sharing group, the structure name and the owning queue manager.

After the header has been completed, the queue manager registers the new shared message data set as active and broadcasts an event to notify any other active queue managers about the new data set.

Every time a queue manager opens a shared message data set it validates the header information to ensure that the correct data set is still being used and that it has not been damaged.

z/OS *Planning your Db2 environment*

If you are using queue sharing groups, IBM MQ needs to attach to a Db2 subsystem that is a member of a data sharing group. Use this topic to help understand the Db2 requirements used to hold IBM MQ data.

IBM MQ needs to know the name of the data sharing group that it is to connect to, and the name of a Db2 subsystem (or Db2 group) to connect to, to reach this data sharing group. These names are specified in the QSGDATA parameter of the CSQ6SYSP system parameter macro (described in Using CSQ6SYSP).

Within the data sharing group, shared Db2 tables are used to hold:

- Configuration information for the queue sharing group.
- Properties of IBM MQ shared and group objects.
- Optionally, data relating to offloaded IBM MQ messages.

IBM MQ provides a single set of sample jobs for defining the necessary Db2 table spaces, tables, and indexes. These jobs make use of Universal Table Spaces (UTS). Earlier versions of the product had two sets of jobs, one for UTS, and one for older types of table space, which have been deprecated by the most recent versions of Db2.

IBM MQ can still be used with older types of table space, and this might be appropriate if you already have an existing queue sharing group. However, if you are creating a new queue sharing group, it should use UTS.

Db2 V12 Function level 508 provides a non disruptive migration process for migrating multi-table table spaces to universal table spaces. You can use this approach to migrate the multi-table table spaces, used by existing queue sharing groups, to universal table spaces without taking an outage of the whole queue sharing group.

In Db2 V13, use the MOVE TABLE option of the ALTER TABLESPACE statement. See Moving tables from multi-table table spaces to partition-by-growth table spaces for more information.

By default Db2 uses the user ID of the person running the jobs as the owner of the Db2 resources. If this user ID is deleted then the resources associated with it are deleted, and so the table is deleted. Consider using a group ID to own the tables, rather than an individual user ID. You can do this by adding `GROUP=groupname` onto the JOB card, and specifying `SET CURRENT SQLID='groupname'` before any SQL statements.

IBM MQ uses the RRS Attach facility of Db2. This means that you can specify the name of a Db2 group that you want to connect to. The advantage of connecting to a Db2 group attach name (rather than a specific Db2 subsystem), is that IBM MQ can connect (or reconnect) to any available Db2 subsystem on the z/OS image that is a member of that group. There must be a Db2 subsystem that is a member of the data sharing group active on each z/OS image where you are going to run a queue-sharing IBM MQ subsystem, and RRS must be active.

Db2 storage

For most installations, the amount of Db2 storage required is about 20 or 30 cylinders on a 3390 device. However, if you want to calculate your storage requirement, the following table gives some information to help you determine how much storage Db2 requires for the IBM MQ data. The table describes the length of each Db2 row, and when each row is added to or deleted from the relevant Db2 table. Use this information together with the information about calculating the space requirements for the Db2 tables and their indexes in the *Db2 for z/OS Installation Guide*.

Table 25. Planning your Db2 storage requirements

Db2 table name	Length of row	A row is added when:	A row is deleted when:
CSQ.ADMIN_B_QSG	252 bytes	A queue sharing group is added to the table with the ADD QSG function of the CSQ5PQSG utility.	A queue sharing group is removed from the table with the REMOVE QSG function of the CSQ5PQSG utility. (All rows relating to this queue sharing group are deleted automatically from all the other Db2 tables when the queue sharing group record is deleted.)
CSQ.ADMIN_B_QMGR	Up to 3828 bytes	A queue manager is added to the table with the ADD QMGR function of the CSQ5PQSG utility.	A queue manager is removed from the table with the REMOVE QMGR function of the CSQ5PQSG utility.
CSQ.ADMIN_B_STRUCTURE	1454 bytes	The first local queue definition, specifying the QSGDISP(SHARED) attribute, that names a previously unknown structure within the queue sharing group is defined.	The last local queue definition, specifying the QSGDISP(SHARED) attribute, that names a structure within the queue sharing group is deleted.
CSQ.ADMIN_B_SCST	342 bytes	A shared channel is started.	A shared channel becomes inactive.
CSQ.ADMIN_B_SSKT	254 bytes	A shared channel that has the NPMSPEED(NORMAL) attribute is started.	A shared channel that has the NPMSPEED(NORMAL) attribute becomes inactive.

Table 25. Planning your Db2 storage requirements (continued)

Db2 table name	Length of row	A row is added when:	A row is deleted when:
CSQ.ADMIN_B_STRBACKUP	514 bytes	A new row is added to the CSQ.ADMIN_B_STRUCTURE table. Each entry is a dummy entry until the BACKUP CFSTRUCT command is run, which overwrites the dummy entries.	A row is deleted from the CSQ.ADMIN_B_STRUCTURE table.
CSQ.OBJ_B_AUTHINFO	3400 bytes	An authentication information object with QSGDISP(GROUP) is defined.	An authentication information object with QSGDISP(GROUP) is deleted.
CSQ.OBJ_B_QUEUE	Up to 3707 bytes	• A queue with the QSGDISP(GROUP) attribute is defined. • A queue with the QSGDISP(SHARED) attribute is defined. • A model queue with the DEFTYPE(SHAREDYN) attribute is opened.	• A queue with the QSGDISP(GROUP) attribute is deleted. • A queue with the QSGDISP(SHARED) attribute is deleted. • A dynamic queue with the DEFTYPE(SHAREDYN) attribute is closed with the DELETE option.
CSQ.OBJ_B_NAMELIST	Up to 15127 bytes	A namelist with the QSGDISP(GROUP) attribute is defined.	A namelist with the QSGDISP(GROUP) attribute is deleted.
CSQ.OBJ_B_CHANNEL	Up to 14127 bytes	A channel with the QSGDISP(GROUP) attribute is defined.	A channel with the QSGDISP(GROUP) attribute is deleted.
CSQ.OBJ_B_STGCLASS	Up to 2865 bytes	A storage class with the QSGDISP(GROUP) attribute is defined.	A storage class with the QSGDISP(GROUP) attribute class is deleted.
CSQ.OBJ_B_PROCESS	Up to 3347 bytes	A process with the QSGDISP(GROUP) attribute is defined.	A process with the QSGDISP(GROUP) attribute is deleted.
CSQ.OBJ_B_TOPIC	Up to 14520 bytes	A topic object with QSGDISP(GROUP) attribute is defined.	A topic object with QSGDISP(GROUP) attribute is deleted.
CSQ.EXTEND_B_QMGR	Less than 430 bytes	A queue manager is added to the table with the ADD QMGR function of the CSQ5PQSG utility.	A queue manager is removed from the table with the REMOVE QMGR function of the CSQ5PQSG utility.
CSQ.ADMIN_B_MESSAGES	87 bytes	For large message PUT (1 per BLOB).	For large message GET (1 per BLOB).

Table 25. Planning your Db2 storage requirements (continued)			
Db2 table name	**Length of row**	**A row is added when:**	**A row is deleted when:**
CSQ.ADMIN_MSGS_BAUX1 CSQ.ADMIN_MSGS_BAUX2 CSQ.ADMIN_MSGS_BAUX3 CSQ.ADMIN_MSGS_BAUX4		These 4 tables contain message payload for large messages added into one of these 4 tables for each BLOB of the message. BLOBS are up to 511 KB in length, so if the message size is > 711 KB, there will be multiple BLOBs for this message.	

The use of large numbers of shared queue messages of size greater than 63 KB can have significant performance implications on your IBM MQ system. For more information, see SupportPac MP16, Capacity Planning and Tuning for IBM MQ for z/OS, at: SupportPacs for IBM MQ and other project areas.

z/OS Planning for backup and recovery

Developing backup and recovery procedures at your site is vital to avoid costly and time-consuming losses of data. IBM MQ provides means for recovering both queues and messages to their current state after a system failure.

This topic contains the following sections:

- "Recovery procedures" on page 184
- "Tips for backup and recovery" on page 185
- "Recovering page sets" on page 187
- "Recovering CF structures" on page 188
- "Achieving specific recovery targets" on page 189
- "Backup considerations for other products" on page 190
- "Recovery and CICS" on page 190
- "Recovery and IMS" on page 190
- "Preparing for recovery on an alternative site" on page 191
- "Example of queue manager backup activity" on page 191

Recovery procedures

Develop the following procedures for IBM MQ:

- Creating a point of recovery.
- Backing up page sets.
- Backing up CF structures.
- Recovering page sets.
- Recovering from out-of-space conditions (IBM MQ logs and page sets).
- Recovering CF structures.

See Administering IBM MQ for z/OS for information about these.

Become familiar with the procedures used at your site for the following:

- Recovering from a hardware or power failure.
- Recovering from a z/OS component failure.

- Recovering from a site interruption, using off-site recovery.

z/OS Tips for backup and recovery

Use this topic to understand some backup and recovery tasks.

The queue manager restart process recovers your data to a consistent state by applying log information to the page sets. If your page sets are damaged or unavailable, you can resolve the problem using your backup copies of your page sets (if all the logs are available). If your log data sets are damaged or unavailable, it might not be possible to recover completely.

Consider the following points:

- Periodically take backup copies
- Do not discard archive logs you might need
- Do not change the DDname to page set association

Periodically take backup copies

A *point of recovery* is the term used to describe a set of backup copies of IBM MQ page sets and the corresponding log data sets required to recover these page sets. These backup copies provide a potential restart point in the event of page set loss (for example, page set I/O error). If you restart the queue manager using these backup copies, the data in IBM MQ is consistent up to the point that these copies were taken. Provided that all logs are available from this point, IBM MQ can be recovered to the point of failure.

The more recent your backup copies, the quicker IBM MQ can recover the data in the page sets. The recovery of the page sets is dependent on all the necessary log data sets being available.

In planning for recovery, you need to determine how often to take backup copies and how many complete backup cycles to keep. These values tell you how long you must keep your log data sets and backup copies of page sets for IBM MQ recovery.

When deciding how often to take backup copies, consider the time needed to recover a page set. The time needed is determined by the following:

- The amount of log to traverse.
- The time it takes an operator to mount and remove archive tape volumes.
- The time it takes to read the part of the log needed for recovery.
- The time needed to reprocess changed pages.
- The storage medium used for the backup copies.
- The method used to make and restore backup copies.

In general, the more frequently you make backup copies, the less time recovery takes, but the more time is spent making copies.

For each queue manager, you should take backup copies of the following:

- The archive log data sets
- The BSDS copies created at the time of the archive
- The page sets
- Your object definitions
- Your CF structures

To reduce the risk of your backup copies being lost or damaged, consider:

- Storing the backup copies on different storage volumes to the original copies.
- Storing the backup copies at a different site to the original copies.

- Making at least two copies of each backup of your page sets and, if you are using single logging or a single BSDS, two copies of your archive logs and BSDS. If you are using dual logging or BSDS, make a single copy of both archive logs or BSDS.

Before moving IBM MQ to a production environment, fully test and document your backup procedures.

Backing up your page sets

You need to back up page sets regularly. Some enterprises back up the page sets twice a day.

You need the active and archive logs since a backup to be able to recover using the backup. You need enough log data to go back four checkpoints if the backup was taken when the queue manager was running.

You can use ADRDSSU FastReplication to back up page sets, and you can do this while the queue manager is active. Note that you need to ensure there is enough space in the storage pool.

Backing up your object definitions

Create backup copies of your object definitions. To do this, use the MAKEDEF feature of the COMMAND function of the utility program (described in Using the COMMAND function of CSQUTIL).

You should do this whenever you take backup copies of your queue manager data sets, and keep the most current version.

Backing up your coupling facility structures

If you have set up any queue sharing groups, even if you are not using them, you must take periodic backups of your CF structures. To do this, use the IBM MQ BACKUP CFSTRUCT command. You can use this command only on CF structures that are defined with the RECOVER(YES) attribute. If any CF entries for persistent shared messages refer to offloaded message data stored in a shared message data set (SMDS) or Db2, the offloaded data is retrieved and backed up together with the CF entries. Shared message data sets should not be backed up separately.

It is recommended that you take a backup of all your CF structures about every hour, to minimize the time it takes to restore a CF structure.

You could perform all your CF structure backups on a single queue manager, which has the advantage of limiting the increase in log use to a single queue manager. Alternatively, you could perform backups on all the queue managers in the queue sharing group, which has the advantage of spreading the workload across the queue sharing group. Whichever strategy you use, IBM MQ can locate the backup and perform a RECOVER CFSTRUCT from any queue manager in the queue sharing group. The logs of all the queue managers in the queue sharing group need to be accessed to recover the CF structure.

Backing up your message security policies

If you are using Advanced Message Security to create a backup of your message security policies, create a backup using the message security policy utility (CSQ0UTIL) to run **dspmqspl** with the -export parameter, then save the policy definitions that are output to the EXPORT DD.

You should create a backup of your message security policies whenever you take backup copies of your queue manager data sets, and keep the most current version.

Do not discard archive logs you might need

IBM MQ might need to use archive logs during restart. You must keep sufficient archive logs so that the system can be fully restored. IBM MQ might use an archive log to recover a page set from a restored backup copy. If you have discarded that archive log, IBM MQ cannot restore the page set to its current state. When and how you discard archive logs is described in Discarding archive log data sets.

You can use the `/cpf DIS USAGE TYPE(ALL)` command to display the log RBA, and log range sequence number (LRSN) that you need to recover your queue manager's page sets and the queue sharing group's structures. You should then use the print log map utility (CSQJU004) to print bootstrap data set (BSDS) information for the queue manager to locate the logs containing the log RBA.

For CF structures, you need to run the CSQJU004 utility on each queue manager in the queue sharing group to locate the logs containing the LRSN. You need these logs and any later logs to be able to recover the page sets and structures.

Do not change the DDname to page set association

IBM MQ associates page set number 00 with DDname CSQP0000, page set number 01 with DDname CSQP0001, and so on, up to CSQP0099. IBM MQ writes recovery log records for a page set based on the DDname that the page set is associated with. For this reason, you must not move page sets that have already been associated with a PSID DDname.

z/OS Recovering page sets

Use this topic to understand the factors involved when recovering pages sets, and how to minimize restart times.

A key factor in recovery strategy concerns the time for which you can tolerate a queue manager outage. The total outage time might include the time taken to recover a page set from a backup, or to restart the queue manager after an abnormal termination. Factors affecting restart time include how frequently you back up your page sets, and how much data is written to the log between checkpoints.

To minimize the restart time after an abnormal termination, keep units of work short so that, at most, two active logs are used when the system restarts. For example, if you are designing an IBM MQ application, avoid placing an MQGET call that has a long wait interval between the first in-syncpoint MQI call and the commit point because this might result in a unit of work that has a long duration. Another common cause of long units of work is batch intervals of more than 5 minutes for the channel initiator.

You can use the DISPLAY THREAD command to display the RBA of units of work and to help resolve the old ones.

How often must you back up a page set?

Frequent page set backup is essential if a reasonably short recovery time is required. This applies even when a page set is very small or there is a small amount of activity on queues in that page set.

If you use persistent messages in a page set, the backup frequency should be in hours rather than days. This is also the case for page set zero.

To calculate an approximate backup frequency, start by determining the target total recovery time. This consists of the following:

1. The time taken to react to the problem.
2. The time taken to restore the page set backup copy.

   If you use SnapShot backup/restore, the time taken to perform this task is a few seconds. For information about SnapShot, see the *DFSMSdss Storage Administration Guide*.
3. The time the queue manager requires to restart, including the additional time needed to recover the page set.

   This depends most significantly on the amount of log data that must be read from active and archive logs since that page set was last backed up. All such log data must be read, in addition to that directly associated with the damaged page set.

   Note: When using *fuzzy backup* (where a snapshot is taken of the logs and page sets while a unit of work is active), it might be necessary to read up to three additional checkpoints, and this might result in the need to read one or more additional logs.

When deciding on how long to allow for the recovery of the page set, the factors that you need to consider are:

- The rate at which data is written to the active logs during normal processing depends on how messages arrive in your system, in addition to the message rate.

  Messages received or sent over a channel result in more data logging than messages generated and retrieved locally.
- The rate at which data can be read from the archive and active logs.

  When reading the logs, the achievable data rate depends on the devices used and the total load on your particular DASD subsystem.

  With most tape units, it is possible to achieve higher data rates for archived logs with a large block size. However, if an archive log is required for recovery, all the data on the active logs must be read also.

z/OS Recovering CF structures

Use this topic to understand the recovery process for CF structures.

At least one queue manager in the queue sharing group must be active to process a RECOVER CFSTRUCT command. CF structure recovery does not affect queue manager restart time, because recovery is performed by an already active queue manager.

The recovery process consists of two logical steps that are managed by the RECOVER CFSTRUCT command:

1. Locating and restoring the backup.
2. Merging all the logged updates to persistent messages that are held on the CF structure from the logs of all the queue managers in the queue sharing group that have used the CF structure, and applying the changes to the backup.

The second step is likely to take much longer because a lot of log data might need to be read. You can reduce the time taken if you take frequent backups, or if you recover multiple CF structures at the same time, or both.

The queue manager performing the recovery locates the relevant backups on all the other queue managers' logs using the data in Db2 and the bootstrap data sets. The queue manager replays these backups in the correct time sequence across the queue sharing group, from just before the last backup through to the point of failure.

The time it takes to recover a CF structure depends on the amount of recovery log data that must be replayed, which in turn depends on the frequency of the backups. In the worst case, it takes as long to read a queue manager's log as it did to write it. So if, for example, you have a queue sharing group containing six queue managers, an hour's worth of log activity could take six hours to replay. In general it takes less time than this, because reading can be done in bulk, and because the different queue manager's logs can be read in parallel. As a starting point, we recommend that you back up your CF structures every hour.

All queue managers can continue working with non-shared queues and queues in other CF structures while there is a failed CF structure. If the administration structure has also failed, at least one of the queue managers in the queue sharing group must be started before you can issue the RECOVER CFSTRUCT command.

Backing up CF structures can require considerable log writing capacity, and can therefore impose a large load on the queue manager doing the backup. Choose a lightly loaded queue manager for doing backups; for busy systems, add an additional queue manager to the queue sharing group and dedicate it exclusively for doing backups.

z/OS Achieving specific recovery targets

Use this topic for guidance on how you can achieve specific recovery target times by adjusting backup frequency.

If you have specific recovery targets to achieve, for example, completion of the queue manager recovery and restart processing in addition to the normal startup time within *xx* seconds, you can use the following calculation to estimate your backup frequency (in hours):

```
Formula (A)
                     Required restart time * System recovery log read rate
                           (in secs)                  (in MB/sec)
Backup frequency  = ------------------------------------------------------
   (in hours)              Application log write rate (in MB/hour)
```

Note: The examples given next are intended to highlight the need to back up your page sets frequently. The calculations assume that most log activity is derived from a large number of persistent messages. However, there are situations where the amount of log activity is not easily calculated. For example, in a queue sharing group environment, a unit of work in which shared queues are updated in addition to other resources might result in UOW records being written to the IBM MQ log. For this reason, the Application log write rate in Formula (A) can be derived accurately only from the observed rate at which the IBM MQ logs fill.

For example, consider a system in which IBM MQ MQI clients generate a total load of 100 persistent messages a second. In this case, all messages are generated locally.

If each message is of user length 1 KB, the amount of data logged each hour is approximately:

```
100 * (1 + 1.3) KB * 3600 = approximately 800 MB

where
      100           = the message rate a second
      (1 + 1.3) KB  = the amount of data logged for
                      each 1 KB of persistent messages
```

Consider an overall target recovery time of 75 minutes. If you have allowed 15 minutes to react to the problem and restore the page set backup copy, queue manager recovery and restart must then complete within 60 minutes (3600 seconds) applying formula (A). Assuming that all required log data is on RVA2-T82 DASD, which has a recovery rate of approximately 2.7 MB a second, this necessitates a page set backup frequency of at least every:

```
3600 seconds * 2.7 MB a second / 800 MB an hour = 12.15 hours
```

If your IBM MQ application day lasts approximately 12 hours, one backup each day is appropriate. However, if the application day lasts 24 hours, two backups each day is more appropriate.

Another example might be a production system in which all the messages are for request-reply applications (that is, a persistent message is received on a receiver channel and a persistent reply message is generated and sent down a sender channel).

In this example, the achieved batch size is one, and so there is one batch for every message. If there are 50 request replies a second, the total load is 100 persistent messages a second. If each message is 1 KB in length, the amount of data logged each hour is approximately:

```
50((2 * (1+1.3) KB) + 1.4 KB + 2.5 KB) * 3600 = approximately 1500 MB

where:
 50                 = the message pair rate a second
 (2 * (1 + 1.3) KB) = the amount of data logged for each message pair
 1.4 KB             = the overhead for each batch of messages
                      received by each channel
 2.5 KB             = the overhead for each batch of messages sent
                      by each channel
```

To achieve the queue manager recovery and restart within 30 minutes (1800 seconds), again assuming that all required log data is on RVA2-T82 DASD, this requires that page set backup is carried out at least every:

```
1800 seconds * 2.7 MB a second / 1500 MB an hour = 3.24 hours
```

Periodic review of backup frequency

Monitor your IBM MQ log usage in terms of MB an hour. Periodically perform this check and amend your page set backup frequency if necessary.

z/OS Backup considerations for other products

If you are using IBM MQ with CICS or IMS then you must also consider the implications for your backup strategy with those products. The data facility hierarchical storage manager (DFHSM) manages data storage, and can interact with the storage used by IBM MQ.

Backup and recovery with DFHSM

The data facility hierarchical storage manager (DFHSM) does automatic space-availability and data-availability management among storage devices in your system. If you use it, you need to know that it moves data to and from the IBM MQ storage automatically.

DFHSM manages your DASD space efficiently by moving data sets that have not been used recently to alternative storage. It also makes your data available for recovery by automatically copying new or changed data sets to tape or DASD backup volumes. It can delete data sets, or move them to another device. Its operations occur daily, at a specified time, and allow for keeping a data set for a predetermined period before deleting or moving it.

You can also perform all DFHSM operations manually. For more information on DFHSM, see the z/OS DFSMS product documentation. If you use DFHSM with IBM MQ, note that DFHSM does the following:

- Uses cataloged data sets.
- Operates on page sets and logs.
- Supports VSAM data sets.

Recovery and CICS

The recovery of CICS resources is not affected by the presence of IBM MQ. CICS recognizes IBM MQ as a non-CICS resource (or external resource manager), and includes IBM MQ as a participant in any syncpoint coordination requests using the CICS resource manager interface (RMI). For more information about CICS recovery and the CICS resource manager interface, see the CICS product documentation.

Recovery and IMS

IMS recognizes IBM MQ as an external subsystem and as a participant in syncpoint coordination. IMS recovery for external subsystem resources is described in the IMS product documentation.

z/OS Preparing for recovery on an alternative site

If a total loss of an IBM MQ computing center, you can recover on another IBM MQ system at a recovery site.

To recover an IBM MQ system at a recovery site, you must regularly back up the page sets and the logs. As with all data recovery operations, the objectives of disaster recovery are to lose as little data, workload processing (updates), and time as possible.

At the recovery site:

- The recovery IBM MQ queue manager **must** have the same name as the lost queue manager.
- Ensure the system parameter module used on the recovery queue manager contains the same parameters as the lost queue manager.

See Administering IBM MQ for z/OS and Troubleshooting IBM MQ for z/OS problems for more information.

z/OS Example of queue manager backup activity

This topic shows as an example of queue manager backup activity.

When you plan your queue manager backup strategy, a key consideration is retention of the correct amount of log data. Managing the logs describes how to determine which log data sets are required, by reference to the system recovery RBA of the queue manager. IBM MQ determines the system recovery RBA using information about the following:

- Currently active units of work.
- Page set updates that have not yet been flushed from the buffer pools to disk.
- CF structure backups, and whether this queue manager's log contains information required in any recovery operation using them.

You must retain sufficient log data to be able to perform media recovery. While the system recovery RBA increases over time, the amount of log data that must be retained only decreases when subsequent backups are taken. CF structure backups are managed by IBM MQ, and so are taken into account when reporting the system recovery RBA. This means that in practice, the amount of log data that must be retained only reduces when page set backups are taken.

Figure 43 on page 192 shows an example of the backup activity on a queue manager that is a member of a queue sharing group, how the recovery RBA varies with each backup, and how that affects the amount of log data that must be retained. In the example the queue manager uses local and shared resources: page sets, and two CF structures, STRUCTURE1 and STRUCTURE2.

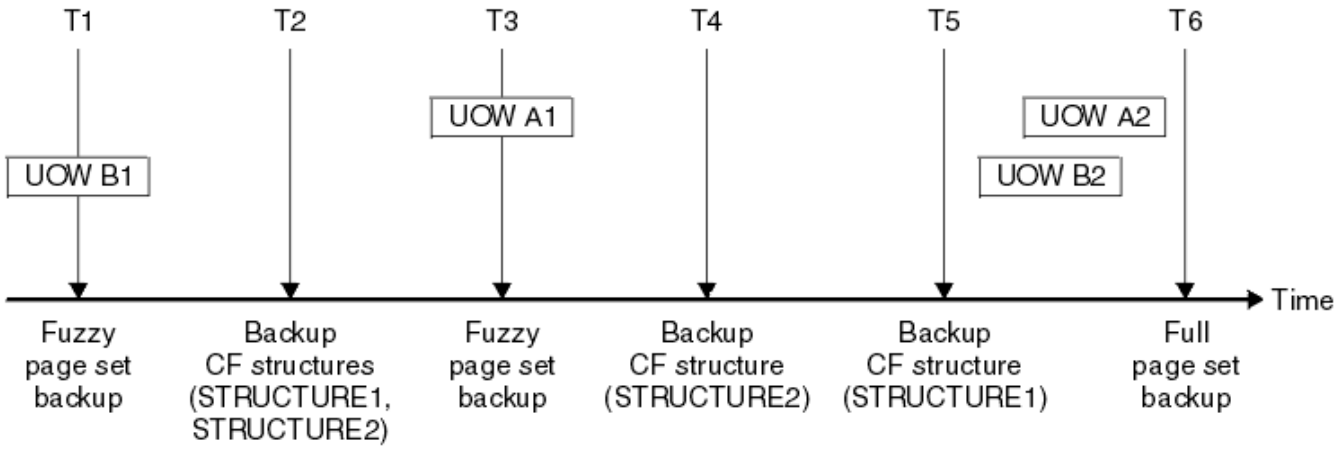


Figure 43. Example of queue manager backup activity

This is what happens at each point in time:

Point in time T1

A fuzzy backup is created of your page sets, as described in How to back up and recover page sets.

The system recovery RBA of the queue manager is the lowest of the following:

- The recovery RBAs of the page sets being backed up at this point.
- The lowest recovery RBA required to recover the CF application structures. This relates to the recovery of backups of STRUCTURE1 and STRUCTURE2 created earlier.
- The recovery RBA for the oldest currently active unit of work within the queue manager (UOWB1).

The system recovery RBA for this point in time is given by messages issued by the DISPLAY USAGE command, which is part of the fuzzy backup process.

Point in time T2

Backups of the CF structures are created. CF structure STRUCTURE1 is backed up first, followed by STRUCTURE2.

The amount of log data that must be retained is unchanged, because the same data as determined from the system recovery RBA at T1 is still required to recover using the page set backups taken at T1.

Point in time T3

Another fuzzy backup is created.

The system recovery RBA of the queue manager is the lowest of the following:

- The recovery RBAs of the page sets being backed up at this point.
- The lowest recovery RBA required to recover CF structure STRUCTURE1, because STRUCTURE1 was backed up before STRUCTURE2.
- The recovery RBA for the oldest currently active unit of work within the queue manager (UOWA1).

The system recovery RBA for this point in time is given by messages issued by the DISPLAY USAGE command, which is part of the fuzzy backup process.

You can now reduce the log data retained, as determined by this new system recovery RBA.

Point in time T4

A backup is taken of CF structure STRUCTURE2. The recovery RBA for the recovery of the oldest required CF structure backup relates to the backup of CF structure STRUCTURE1, which was backed up at time T2.

The creation of this CF structure backup has no effect on the amount of log data that must be retained.

Point in time T5

A backup is taken of CF structure STRUCTURE1. The recovery RBA for recovery of the oldest required CF structure backup now relates to recovery of CF structure STRUCTURE2, which was backed up at time T4.

The creation of this CF structure backup has no effect on amount of log data that must be retained.

Point in time T6

A full backup is taken of your page sets as described in How to back up and recover page sets.

The system recovery RBA of the queue manager is the lowest of the following:

- The recovery RBAs of the page sets being backed up at this point.
- The lowest recovery RBA required to recover the CF structures. This relates to recovery of CF structure STRUCTURE2.
- The recovery RBA for the oldest currently active unit of work within the queue manager. In this case, there are no current units of work.

The system recovery RBA for this point in time is given by messages issued by the DISPLAY USAGE command, which is part of the full backup process.

Again, the log data retained can be reduced, because the system recovery RBA associated with the full backup is more recent.

z/OS Planning your z/OS UNIX environment

Certain processes within the IBM MQ queue manager, channel initiator, and mqweb server use z/OS UNIX System Services (z/OS UNIX) for their normal processing.

The queue manager and channel initiator started task user IDs need an OMVS segment with a UID defined in order to be able to access z/OS UNIX. The user IDs require no special permissions in z/OS UNIX.

Note: Although the queue manager and channel initiator make use of z/OS UNIX facilities (for example, to interface with TCP/IP services), they do not need to access any of the content of the IBM MQ installation directory in the z/OS UNIX file system. As a result, the queue manager and channel initiator do not require any configuration to specify the path for the z/OS UNIX file system.

The mqweb server, which hosts the IBM MQ Console and REST API, makes use of files in the IBM MQ installation directory in the z/OS UNIX file system. It also needs access to another file system which is used to store data such as configuration and log files. The mqweb started task JCL needs to be customized to reference these z/OS UNIX file systems.

The content of the IBM MQ directory in the z/OS UNIX file system is also used by applications connecting to IBM MQ. For example, applications using the IBM MQ classes for Java or IBM MQ classes for JMS interfaces.

See the following topics for the relevant configuration instructions:

- Environment variables relevant to IBM MQ classes for Java
- IBM MQ classes for Java libraries
- Setting environment variables
- Configuring the Java Native Interface (JNI) libraries

z/OS Planning for Advanced Message Security on z/OS

TLS (or SSL) can be used to encrypt and protect messages flowing on a network, but this does not protect messages when they are on a queue ("at rest"). Advanced Message Security (AMS) protects the messages from the time that they are first put to a queue, until they are got, so that only the intended recipients of the message can read that message. The messages are encrypted and signed during put processing, and unprotected during get processing.

AMS can be configured to protect messages in different ways:

1. A message can be signed. The message is in clear text, but there is a checksum, which is signed. This allows any changes in the message content to be detected. From the signed content, you can identify who signed the data.
2. A message can be encrypted. The contents are not visible to anyone without the decryption key. The decryption key is encrypted for each recipient.
3. A message can be encrypted and signed. They decryption key is encrypted for each recipient, and from the signing you can identify who sent the message.

The encryption and signing use digital certificates and key rings.

You can set up a client to use AMS, so the data is protected before the data is put on the client channel. Protected messages can be sent to a remote queue manager, and you need to configure the remote queue manager to process these messages.

Setting up AMS

An AMS address space is used for doing the AMS work. This has additional security set up, to give access to and protect the use of key rings and certificates.

You configure which queues are to be protected by using a utility program (CSQ0UTIL) to define the security policies for queues.

Once AMS is set up

You need to set up a digital certificate and a key ring for people who put messages, and the people who get messages.

If a user, Alice, on z/OS needs to send a message to Bob, AMS needs a copy of the public certificate for Bob.

If Bob wants to process a message from Alice, AMS needs the public certificate for Alice, or the same certificate authority certificate used by Alice.

Attention: You need to:

- Carefully plan who can put to, or get from, queues
- Identify the people and their certificate names.

It is easy to make mistakes, and problems can be hard to resolve.

Related concepts

“Planning for your queue manager” on page 138
When you are setting up a queue manager, your planning should allow for the queue manager to grow, so that the queue manager meets the needs of your enterprise.

z/OS Planning for Managed File Transfer on z/OS

Use this section as guidance on how you need to set up your system to run Managed File Transfer (MFT) on z/OS.

z/OS Planning for Managed File Transfer - hardware and software requirements on z/OS

Use this topic as guidance on how you need to set up hardware and software requirements on your system to run Managed File Transfer (MFT) on z/OS.

Software requirements

Managed File Transfer is written in Java, with some shell scripts and JCL to configure and operate the program.

Important: You must be familiar with z/OS UNIX System Services (z/OS UNIX) in order to configure Managed File Transfer. For example:

- The file directory structure, with names such as `/u/userID/myfile.txt`
- z/OS UNIX commands, for example:
  - `cd` (change directory)
  - `ls` (list)
  - `chmod` (change the file permissions)
  - `chown` (change file ownership or groups which can access the file or directory)

You require the following products in z/OS UNIX to be able to configure and run MFT:

1. Java, for example, in directory `/java/java80_bit64_GA/J8.0_64/`
2. IBM MQ 10.0.0, for example, in directory `/mqm/V10R0M0`
3. If you want to use Db2 for status and history, you need to install Db2 JDBC libraries, for example, in directory `/db2/db2v10/jdbc/libs`.

Product registration

At startup Managed File Transfer checks the registration in `sys1.parmlib`(IFAPRDxx) concatenation. The following code is an example of how you register MFT:

```
PRODUCT OWNER('IBM CORP')
NAME('WS MQ FILE TRANS')
ID(5655-MFT)
VERSION(*) RELEASE(*) MOD(*)
FEATURENAME('WS MQ FILE TRANS')
STATE(ENABLED)
```

Disk space

The IBM MQ for z/OS Program Directory states the DASD and zFS storage requirements for Managed File Transfer. For download links for the Program Directory for IBM MQ for z/OS, see IBM MQ 10.0 PDF files for product documentation and Program Directories.

z/OS Planning for Managed File Transfer - topologies on z/OS

Use this topic as guidance on what topology you need on your system to run Managed File Transfer (MFT) on z/OS.

Managed File Transfer queue managers

IBM MQ Managed File Transfer topologies consist of:

Agents, and their associated queue managers
: The agent uses system queues hosted on their agent queue manager to maintain state information and receive requests for work.

A command queue manager
: This acts as a gateway into an MFT topology. It is connected to the agent queue managers through either sender and receiver channels, or clustering. When certain commands are run, they connect directly to the command queue manager, and send a message to the specified agent. This message is routed through the IBM MQ network to the agent queue manager, where it is picked up by the agent and processed.

A coordination queue manager
: This is a central hub that has knowledge of the entire topology. The coordination queue manager is connected to all of the agent queue managers in a topology through either sender and receiver channels, or using clustering. Agents regularly publish status information to the coordination queue manager, and store their transfer templates there.

It is possible for a single queue manager to perform multiple roles within a topology. For example, the same queue manager can be configured as both the coordination queue manager and the command queue manager for a topology.

If you are using multiple queue managers you need to set up channels between the queue managers. You can either do this by using clustering or by using point-to-point connections.

When using IBM MQ Managed File Transfer for z/OS, there are a number of things to consider when determining which queue managers to use for the different roles within a topology.

Agent queue managers

The agent queue manager for an IBM MQ Managed File Transfer for z/OS agent must be running on z/OS.

If:

- The agent is running Managed File Transfer for z/OS on IBM MQ 9.1 or later
- And, the agent queue manager is licensed for IBM MQ Advanced for z/OS Value Unit Edition (Advanced VUE)

the agent can connect to the queue manager using the CLIENT transport.

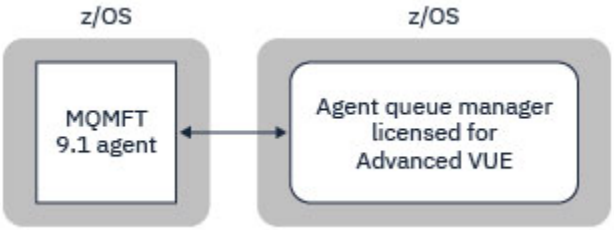


Figure 44. MFT 9.1 agents on z/OS can connect to a queue manager using the CLIENT transport, assuming the queue manager is licensed for Advanced VUE.

If:

- The agent is running Managed File Transfer for z/OS on IBM MQ 9.0 or earlier
- Or, the agent queue manager is running Managed File Transfer for z/OS on IBM MQ 9.0 or later, and the agent queue manager is licensed for either MFT, IBM MQ Advanced for z/OS, or Advanced VUE

the agent must connect to the queue manager using the BINDINGS transport.

Figure 45. MFT 9.0 agents on z/OS and 9.1 agents that have an agent queue manager licensed for either MFT or IBM MQ Advanced, must connect using the BINDINGS transport.

Command queue managers

The Which MFT commands and processes connect to which queue manager topic shows all of the commands that connect to the command queue manager for a Managed File Transfer topology.

Note: When running these commands on z/OS, the command queue manager must also be on z/OS.

If the command queue manager is licensed for Advanced VUE, the commands can connect to the queue manger using the CLIENT transport. Otherwise, the commands must connect to the command queue manager using the BINDINGS transport.

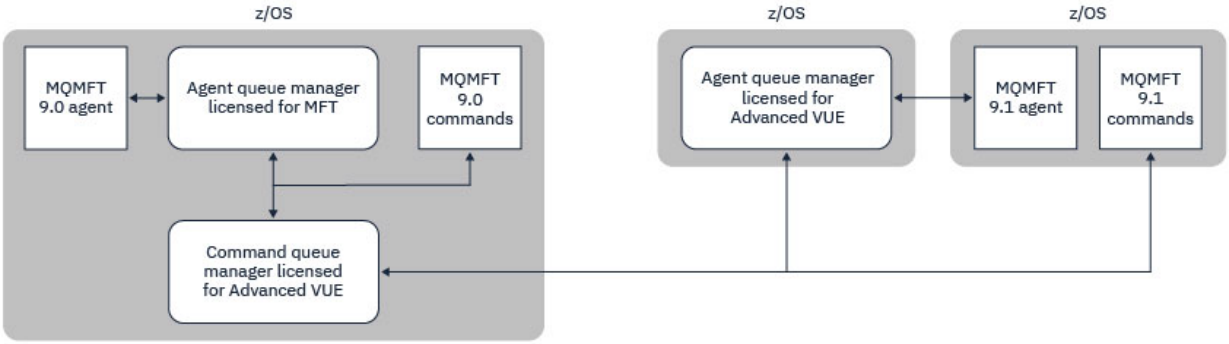


Figure 46. Commands connect to the command queue manager for an MFT topology. When running these commands on z/OS, the command queue manager must also be on z/OS

Coordination queue managers

IBM MQ Managed File Transfer for z/OS agents can be part of a topology where the coordination queue manager is either running on z/OS, or is running on a multiplatform.

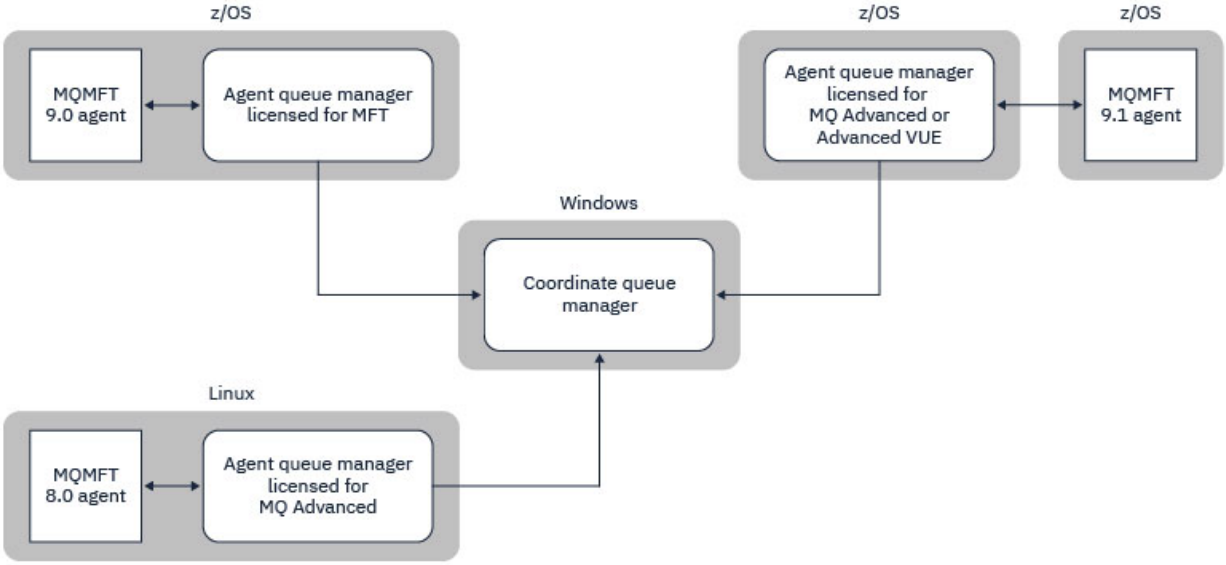


Figure 47. MFT agents running on z/OS can be part of an MFT topology where the coordination queue manager is running on an IBM MQ multiplatform.

The Which MFT commands and processes connect to which queue manager topic shows the commands that connect to the coordination queue manager for a Managed File Transfer topology. It is possible to run these commands on z/OS and have then connect to the coordination queue manager running on a different platform.

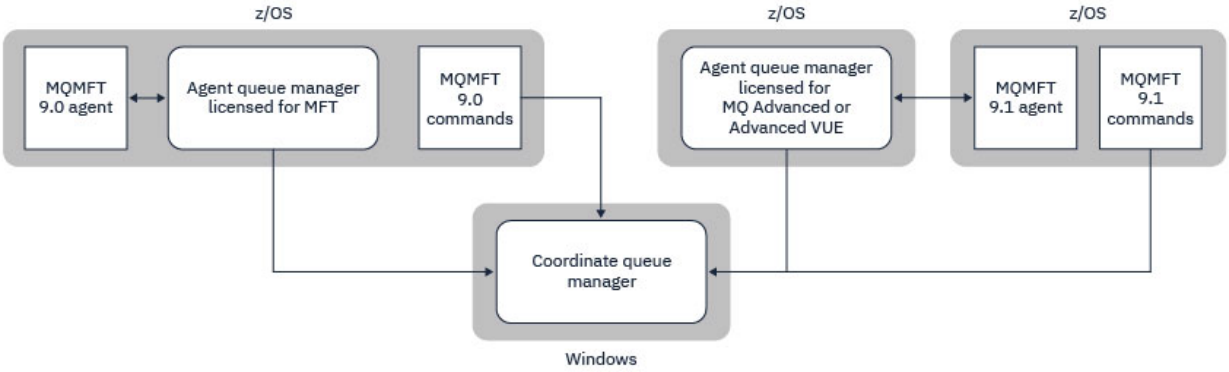


*Figure 48. Certain commands, such as **fteListAgents**, connect directly to the coordination queue manager for an MFT topology.*

How many agents do I need?

The agents do the work in transferring data, and when you make a request to transfer data you specify the name of an agent.

By default an agent can process 25 send and 25 receive requests concurrently. You can configure these processes. See Managed File Transfer configuration options on z/OS for more information.

If the agent is busy then work is queued. The time taken to process a request depends on multiple factors, for example, the amount of data to be sent, the network bandwidth, and the delay on the network.

You might want to have multiple agents to process work in parallel.

You can also control which resources an agent can access, so you might want some agents to work with a limited subset of data.

If you want to process requests with different priority you can use multiple agents and use workload manager to set the priority of the jobs.

Running the agents

Typically the agents are long running processes. The processes can be submitted as jobs that run in batch, or as started tasks.

z/OS Planning for Managed File Transfer - security considerations on z/OS

Use this topic as guidance on what security considerations you need on your system to run Managed File Transfer (MFT) on z/OS.

Security

You need to identify which user IDs are going to be used for MFT configuration and for MFT operation.

You need to identify the files or queues you transfer, and which user IDs are going to be submitting transfer requests to MFT.

When you customize the agents and logger, you specify the group of users that is allowed to run MFT services, or do MFT administration.

You should set up this group before you start customizing MFT. As MFT uses IBM MQ queues, if you have security enabled in the queue manager, MFT requires access to the following resources:

Table 26. MQADMIN resource class	
Name	**Access required**
QUEUE.SYSTEM.FTE.EVENT.agent_name	Update
QUEUE.SYSTEM.FTE.COMMAND.agent_name	Update
CONTEXT.SYSTEM.FTE.COMMAND.agent_name	Update
QUEUE.SYSTEM.FTE.STATE.agent_name	Update
QUEUE.SYSTEM.FTE.DATA.agent_name	Update
QUEUE.SYSTEM.FTE.REPLY.agent_name	Update
QUEUE.SYSTEM.FTE.AUTHAGT1.agent_name	Update
QUEUE.SYSTEM.FTE.AUTHTRN1.agent_name	Update
QUEUE.SYSTEM.FTE.AUTHOPS1.agent_name	Update
QUEUE.SYSTEM.FTE.AUTHSCH1.agent_name	Update
QUEUE.SYSTEM.FTE.AUTHMON1.agent_name	Update
QUEUE.SYSTEM.FTE.AUTHADM1.agent_name	Update

Table 27. MQQUEUE resource class	
Name	**Access required**
SYSTEM.FTE.AUTHAGT1.agent_name	Update
SYSTEM.FTE.AUTHTRN1.agent_name	Update
SYSTEM.FTE.AUTHOPS1.agent_name	Update
SYSTEM.FTE.AUTHSCH1.agent_name	Update
SYSTEM.FTE.AUTHMON1.agent_name	Update

You can use user sandboxing to determine which parts of the file system the user who requests the transfer can access.

To enable user sandboxing, add the `userSandboxes=true` statement to the *agent.properties* file for the agent that you want to restrict, and add appropriate values to the `MQ_DATA_PATH/mqft/config/coordination_qmgr_name/agents/agent_name/UserSandboxes.xml` file.

See Working with user sandboxes for further information.

This user ID is configured in `UserSandboxes.xml` files.

This XML file has information like user ID, or user ID* and a list of resource that can be used (included), or cannot be used (excluded). You need to define specific user IDs that can access which resources: for example:

Table 28. Example user ID together with access to specific resources			
User ID	**Access**	**Include or Exclude**	**Resource**
Admin*	Read	Include	`/home/user/**`
Admin*	Read	Exclude	`/home/user/private/**`
Sysprog	Read	Include	`/home/user/**`
Admin*	Read	Include	Application.reply.queue

Notes:

1. If `type=queue` is specified, the resource is either a queue name, or `queue@qmgr`.
2. If the resource begins with `//`, the resource is a data set; otherwise the resource is a file in z/OS UNIX.
3. The user ID is the user ID from the MQMD structure, so this might not reflect the user ID that actually puts the message.
4. For requests on the local queue manager you can use MQADMIN CONTEXT.* to limit which users can set this value.
5. For requests coming in over a remote queue manager, you have to assume that the distributed queue managers have security enabled to prevent unauthorized setting of the user ID in the MQMD structure.
6. A user ID of SYSPROG1 on a Linux machine, is the same user ID SYSPROG1 for the security checking on z/OS.

z/OS Planning to use the IBM MQ Console and REST API on z/OS

The IBM MQ Console and REST API are applications that run in a WebSphere Liberty (Liberty) server known as mqweb. The mqweb server runs as a started task. The IBM MQ Console allows a web browser to be used to administer queue managers. The REST API provides a simple programmatic interface for applications to do queue manager administration, and to perform messaging.

Installation and configuration files

You need to install the IBM MQ for z/OS UNIX System Services Web Components feature, which will install the files needed to run the mqweb server in z/OS UNIX System Services (z/OS UNIX). You need to be familiar with z/OS UNIX to be able to configure and manage the mqweb server.

See IBM MQ for z/OS Program Directory PDF files for information on installing IBM MQ for z/OS UNIX System Services Components.

The IBM MQ files in z/OS UNIX are installed with various attributes set that are required for the correct operation of the mqweb server. If you need to copy the IBM MQ z/OS UNIX installation files, for example if you have installed IBM MQ on one system, and run IBM MQ on a different system, you should copy the IBM MQ ZFS created during the installation, and mount it read only at the destination. Copying the files in other ways might cause some file attributes to be lost.

You need to decide upon the location for, and create, a Liberty user directory when you create the mqweb server. This directory contains configuration and log files, and the location can be something similar to `/var/mqm/mqweb`.

Using the IBM MQ Console and REST API with queue managers at different levels

The REST API can directly interact only with queue managers that run at the same Version, Release, and Modification (VRM) as the mqweb server which runs the REST API. For example, the IBM MQ 10.0.0 REST API can directly interact only with local queue managers at IBM MQ 10.0.0, and the IBM MQ 9.3.5 REST API can directly interact only with local queue managers at IBM MQ 9.3.5.

You can use the REST API to administer a queue manager at a different version from the mqweb server by configuring a gateway queue manager. However, you need at least one queue manager at the same version as the mqweb server to act as the gateway queue manager. For more information, see Remote administration using the REST API.

The IBM MQ Console can be used to manage local queue managers that run at the same version as the IBM MQ Console. From IBM MQ 9.3.0, you can also use the IBM MQ Console to administer a queue manager running on a remote system, or at a different version to the IBM MQ Console. For more information, see IBM MQ Console: Adding a remote queue manager.

Migration

If you have only one queue manager, you can run the mqweb server as a single started task, and change the libraries it uses when you migrate your queue manager.

If you have more than one queue manager, during migration you can start mqweb servers at different versions by using started tasks with different names. These names can be any name you want. For example, you can start an IBM MQ 9.3.0 mqweb server using a started task named MQWB0930, and an IBM MQ 9.3.5 mqweb server using a started task named MQWB0935.

Then, when you migrate the queue managers from one version to a later version, the queue managers become available in the mqweb server for the later version, and are no longer available in the mqweb server for the earlier version.

After you have migrated all the queue managers to the later version, you can delete the mqweb server for the earlier version.

HTTP ports

The mqweb server uses up to two ports for HTTP:

- One for HTTPS, with a default value of 9443.
- One for HTTP. HTTP is not enabled by default, but if enabled, has a default value of 9080.

If the default port values are in use, you must allocate other ports. If you have more than one mqweb server running simultaneously for more than one version of IBM MQ, you must allocate separate ports for each version. For more information on setting the ports that the mqweb server uses, see Configuring the HTTP and HTTPS ports.

You can use the following TSO command to display information about a port:

```
NETSTAT TCP tcpip (PORT portNumber)
```

where *tcpip* is the name of the TCP/IP address space, and *portNumber* specifies the number of the port to display information about.

Security - starting the mqweb server

The mqweb server user ID needs certain authorities. For more information, see Authority required by the mqweb server started task user ID.

Security - using the IBM MQ Console and REST API

When you use the IBM MQ Console and REST API, you must authenticate as a user that is included in a configured registry. These users are assigned specific roles that determine the actions the users can perform. For example, to use the messaging REST API, a user must be assigned the `MQWebUser` role. For more information about the available roles for the IBM MQ Console and REST API, and the access that these roles grant, see Roles on the IBM MQ Console and REST API.

For more information about configuring security for the IBM MQ Console and REST API, see IBM MQ Console and REST API security.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

© Copyright IBM Corp. 2007, 2026

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

Programming interface information, if provided, is intended to help you create application software for use with this program.

This book contains information on intended programming interfaces that allow the customer to write programs to obtain the services of IBM MQ.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Important: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks

IBM, the IBM logo, ibm.com®, are trademarks of IBM Corporation, registered in many jurisdictions worldwide. A current list of IBM trademarks is available on the Web at "Copyright and trademark information"www.ibm.com/legal/copytrade.shtml. Other product and service names might be trademarks of IBM or other companies.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

This product includes software developed by the Eclipse Project (https://www.eclipse.org/).

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Part Number:

(1P) P/N: