MD16: MQSeries for OS/390 - Getting the best from MQSeries for OS/390

Document:	MD16 SCRIPT
Issued:	10th April 2000
Revision Date:	10th April 2000
Previous Revision Date:	None
Next Review:	As required

Colin Paice
IBM United Kingdom Laboratories
Mail Point 204
Hursley Park
Winchester
England

Take Note!

Before using this SupportPac, be sure to read the general information under "Notices".

First Edition, April 2000

This edition applies to Version 1.0 of MD16: MQSeries for OS/390 - Getting the best from MQSeries for OS/390, and to all subsequent releases and modifications until otherwise indicated in new editions.

A form for reader's comments is provided at the back of this publication. If the form has been removed, address your comments to:

IBM United Kingdom Laboratories
AIM Worldwide Technical Sales (MP102)
IBM United Kingdom Laboratories
Hursley Park
WINCHESTER
Hampshire, SO21 2JN, England

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you. You may continue to use the information that you supply.

Contents

Notices

Summary of Changes

Date	**Changes**
April 2000	Initial release

Bibliography

- *MQSeries for OS/390 Version 2 Release 1 System Management Guide SC34-5374*
- *MQSeries Command Reference SC33-1369*
- *MQSeries SupportPac MP15: MQSeries for OS/390 - Printing statistics and accounting records*
- *MQSeries SupportPac MD01: MQSeries - Standards and conventions*

Introduction

“Installing” MQSeries is just part of implementing MQSeries in your organisation.

This document is for people designing solutions and applications using MQSeries, including application architects and MQSeries administrators. It lists some of the tasks that have to be done to ensure a successful MQSeries implementation.

This document presents ideas and techniques that have been used successfully in customer systems. The list is not complete, nor is it definitive, as there may be many equally valid ways of performing a task. If you find this document useful please tell me, and if sufficient people find it useful I will try to update it with other tips and techniques. If no-one uses it then it will be unlikely to be enhanced.

If you have any tips or techniques please tell me and I will try to incorporate them.

Please send any suggestions for improvements, or additional items to Paice@uk.ibm.com.

How to use this document

The document goes from the planning stage through to ongoing activities. The roles of people involved in implementing MQSeries have also been identified, and the tasks for these people have been identified.

Stages in implementation

Roles and responsibilities

Detailed roles

The following roles are typical of the activities in many customer's OS/390 systems environment. The person in the role should review the chapters listed and decide what action they need to take, if any.

The list can be used as a checklist of activities that need to be done.

System architect

__ • "Strategic planning" on page 5
__ • "Decide how many queue managers are needed and their names" on page 8
__ • "Decide naming convention for object definitions" on page 11
__ • "Decide a data set backup strategy" on page 15
__ • "Monitoring your systems" on page 17
__ • "Do you need accounting information?" on page 17
__ • "Decide on a security strategy" on page 18
__ • "Decide if you want to use MQSeries events" on page 19
__ • "Decide on a security strategy" on page 18
__ • "What level of availability do you need?" on page 19
__ • "Planning for applications" on page 24
__ • "Application design question" on page 25
__ • "What do you plan to do if you lose a queue manager data set?" on page 26
__ • "What testing do you need to do?" on page 27

Product installer

__ • "Strategic planning" on page 5
__ • "Decide naming convention for MQSeries executable library names" on page 9

OS/390 systems programmer

__ • "Decide how many queue managers are needed and their names" on page 8
__ • "Decide naming convention for MQSeries executable library names" on page 9
__ • "Define the priority of the MQSeries address spaces" on page 15
__ • "Decide on a security strategy" on page 18
__ • "What level of availability do you need?" on page 19
__ • "Open edition setup" on page 19
__ • "What do you plan to do if you lose a queue manager data set?" on page 26
__ • "What testing do you need to do?" on page 27

Security administrator

__ • "Decide how many queue managers are needed and their names" on page 8
__ • "Decide naming convention for MQSeries executable library names" on page 9
__ • "Decide naming convention for data sets required for each queue manager" on page 10
__ • "Decide on a security strategy" on page 18
__ • "Tasks that should be done on a regular basis" on page 27

MQSeries Systems programmer

__ • "Decide how many queue managers are needed and their names" on page 8
__ • "Decide naming convention for MQSeries executable library names" on page 9
__ • "Decide naming convention for data sets required for each queue manager" on page 10
__ • "Decide naming convention for object definitions" on page 11
__ • "Decide naming convention for storage classes" on page 11
__ • "Decide naming convention for channels" on page 12
__ • "Decide naming convention for storage classes" on page 11
__ • "Decide naming convention for IMS bridge storage classes" on page 11
__ • "Decide naming convention for queues" on page 11
__ • "Decide naming convention for system parameter modules" on page 12
__ • "Decide where to archive logs" on page 12
__ • "Decide number, size and placement of log data sets" on page 12

__ • “Deleting redundant archive log data sets” on page 13
__ • “Decide the number and size of page sets” on page 13
__ • “Mapping page sets to buffer pools” on page 14
__ • “Define the priority of the MQSeries address spaces” on page 15
__ • “Decide a data set backup strategy” on page 15
__ • “Monitoring your systems” on page 17
__ • “Do you need accounting information?” on page 17
__ • “Decide on a security strategy” on page 18
__ • “Decide if you want to use MQSeries events” on page 19
__ • “What level of availability do you need?” on page 19
__ • “CHINIT and channel setup” on page 21
__ • “TCP/IP set up” on page 22
__ • “APPC set up” on page 23
__ • “Other channel considerations” on page 23
__ • “What do you plan to do if you lose a queue manager data set?” on page 26
__ • “What testing do you need to do?” on page 27
__ • “Tasks that should be done on a regular basis” on page 27
__ • “Develop operational procedures” on page 28

MQSeries Administrator

__ • “Strategic planning” on page 5
__ • “Decide naming convention for object definitions” on page 11
__ • “Decide naming convention for queues” on page 11
__ • “Decide naming convention for channels” on page 12
__ • “Decide naming convention for system parameter modules” on page 12
__ • “Decide if you want to use MQSeries events” on page 19
__ • “CHINIT and channel setup” on page 21
__ • “What do you plan to do if you lose a queue manager data set?” on page 26
__ • “What testing do you need to do?” on page 27
__ • “Tasks that should be done on a regular basis” on page 27

IMS Systems programmer

__ • “Decide naming convention for storage classes” on page 11

TCP/IP Systems programmer

__ • “Strategic planning” on page 5
__ • “Decide how many queue managers are needed and their names” on page 8
__ • “Decide naming convention for channels” on page 12
__ • “Open edition setup” on page 19
__ • “CHINIT and channel setup” on page 21
__ • “TCP/IP set up” on page 22

VTAM Systems programmer

__ • “Strategic planning” on page 5
__ • “Decide how many queue managers are needed and their names” on page 8
__ • “Decide naming convention for channels” on page 12
__ • “CHINIT and channel setup” on page 21
__ • “APPC set up” on page 23

Application architect

__ • "Strategic planning" on page 5
__ • "Decide how many queue managers are needed and their names" on page 8
__ • "Decide naming convention for object definitions" on page 11
__ • "Decide naming convention for queues" on page 11
__ • "Decide the number and size of page sets" on page 13
__ • "Do you need accounting information?" on page 17
__ • "Decide if you want to use MQSeries events" on page 19
__ • "CHINIT and channel setup" on page 21
__ • "Planning for applications" on page 24
__ • "Application design question" on page 25
__ • "What do you plan to do if you lose a queue manager data set?" on page 26
__ • "What testing do you need to do?" on page 27
__ • "Develop operational procedures" on page 28

Data Manager

__ • "Decide naming convention for MQSeries executable library names" on page 9
__ • "Decide where to archive logs" on page 12
__ • "Decide naming convention for data sets required for each queue manager" on page 10
__ • "Create JCL to backup up page sets" on page 15
__ • "Allocate data sets to contain JCL for a queue manager" on page 17
__ • "Decide number, size and placement of log data sets" on page 12
__ • "Deleting redundant archive log data sets" on page 13
__ • "Decide the number and size of page sets" on page 13
__ • "Decide a data set backup strategy" on page 15
__ • "Create JCL to backup up page sets" on page 15
__ • "Decide on a security strategy" on page 18
__ • "Tasks that should be done on a regular basis" on page 27

Operations staff

__ • "Automated processing of MQSeries messages" on page 27
__ • "Review new messages for inclusion in automation" on page 28
__ • "What should Operations staff monitor?" on page 28
__ • "Tasks that should be done on a regular basis" on page 27
__ • "Develop operational procedures" on page 28

Strategic planning

You need to have an MQSeries architect who is responsible for designing the MQSeries architecture across your organisation and operating systems. You also need to have an MQSeries Administrator who implements this architecture and manages such things as the day to day changes in definitions.

The architect needs to work with:

- The applications programmers to decide how the applications will use MQSeries
- The MQSeries administrator to ensure that MQSeries systems are implemented to the agreed architecture, and that problems are identified early
- Systems programmers to estimate what resources will be required
- MQSeries architects outside the organisation, for example other parts of the same company, or with different companies where you use MQSeries to communicate with.
- Security staff to ensure that the appropriate security is used.

You need to anticipate how your usage of MQSeries will change over time. For example, it is easier operationally to plan for multiple queue managers from the beginning, rather than plan for one queue manager and then have to add new queue managers later.

You should plan for 6 months or a year ahead and estimate what your MQSeries usage will be in these time frames. You should also put in place processes to compare your planned MQSeries usage against your actual MQSeries usage.

MQSeries Architect's tasks

There are many areas that the MQSeries architect needs to consider when deciding how MQSeries will be used. Some examples are listed below

1. Decide what other products you need

   There are many components you may need for your applications, and you should identify these before you plan your MQSeries systems, as some of these may not be available on all the platforms you need. For example, you should identify any MQSeries products, integration components, databases, and management tools. These can use significant CPU resources depending on how you set them up, and you should allow for this when estimating the resources used by the MQSeries applications, the queue manager and the channel initiator (CHINIT).

   IBM products include:

   a. MQSeries Integrator, a powerful message-brokering software that provides intelligent rules-based message routing.
   b. MQSeries Workflow, which simplifies integration across the whole enterprise by automating business processes involving people and applications.
   c. Tivoli Manager for MQSeries, which provides systems management facilities.

   Products and tools provided by other companies, and the products of the IBM business partners are listed at http://www.ibm.com/software/mqseries/directory.

   There are documents and free software available from the MQSeries SupportPac library at http://www.ibm.com/software/mqseries/txppacs.

2. Your MQSeries topology.

   For example, you can have a two-tier solution where your distributed nodes come directly into the host queue manager, or you can have a three-tier solution where your distributed nodes connect to an intermediate queue manager, and the intermediate queue manager connects to the host queue manager.

The major benefit of a three-tier solution is that less CPU is required on the host, as less channels are used, and you may get a much larger batch size on these channels, compared to the two-tier case where you have many channels, and may only achieve a batch size of one. The CPU per message for a batch size of 1, is approximately twice the cost compared to an achieved batch size of 3.

The major benefit of a two-tier solution is that the round trip time is usually shorter, and there is no intermediate node to manage.

You should, where possible, use clustering queue mangers introduced in MQSeries for OS/390 V2.1 as this significantly reduces the amount of definitions that you need to provide, and provides workload balancing.

3. How many queue managers you will need, and on which OS/390 systems they will be run on.

   You should consider how many production, and test systems you need from operational and from capacity perspectives.

4. You need to plan for problems. You need to be able to:

   a. Detect a problem
   b. Tolerate the problem until it is resolved
   c. Resolve the problem

   For example:

   a. If a link to a remote queue manager is not working then messages will build up on the transmission queue.

      1) How will you detect this. You could use the MQSeries events, or automate the messages on the console.

      2) Will you allow applications to continue putting messages, or will you specify a maximum depth so application programs get notified that the queue is full. In this case what will the application programs do to handle this event.

   b. What time period can you tolerate a channel not being able to connect to a remote queue manager. You will need to develop procedures for handling this.

5. You need a disaster recovery strategy. For example, if you lose your main data centre, do you want to recover today's data or do you want to make MQSeries available as soon as possible. To recover today's data means having one of your log copies on DASD at a remote site. If you want to make the queue manager available as soon as possible then you will need to restore from data set backups, or use a “new system” with the object definitions from the production system. In both cases you may need to implement operational procedures to get remote queue managers to work with the backup queue manager.

6. You need a strategy for applying service to your MQSeries system, and installing new systems. You might decide to have a test MQSeries to validate application changes, and any product fixes.

   You need to have a process in place to be able to apply urgent fixes, if required, across all of your systems and environments.

7. You need to consider any Service Level Agreements, the message rate you expect to get, and end user response time. How will you determine these when your applications are in production.

8. What queue manager restart time can you tolerate?

   If the queue manager does not end cleanly then there is likely to be an increased restart time while work associated with persistent messages is recovered or backed out. You need to decide the maximum MQSeries restart time you can tolerate.

   You need to test the restart time after the queue manager ends abnormally.

   If you find applications are causing a long restart time, for example, by having long units of work, you should fix the applications. If the applications are well behaved, then for a lot of customers the restart time is acceptable. If the restart time is too long you should consider having multiple queue managers - but you need to test this before you make this decision. You might want to have application design guidelines of short units of work.

9. If there are very long units of work outstanding at startup, you are asked if you wish to commit each unit of work or let the system back it out. If you decide to commit the data then there is risk that a request in two phase commit becomes inconsistent. You need to decide between a faster restart time or the risk of inconsistent data. See “Startup messages when long unit of work are present” on page 27.

   If you have very long units of work you have to reply to the messages before the system will restart.

10. Ensure there are no system affinities for your applications and configuration. You need to be prepared to be able to start your application suites (including CICS, DB2, IMS as well as MQSeries) on different images in a sysplex.

The author would like feedback on how useful this book is so if you have found this chapter or the document useful please send an email saying so to Paice@uk.ibm.com. If you are willing to share your experiences on any topic in this document please send them, and the author will try to incorporate them in a future draft.

Planning the MQSeries customisation

Naming conventions

See also SupportPac “MD01: MQSeries - Standards and conventions” available from
http://www.ibm.com/software/mqseries/txppacs/md01.html.

You need to have a naming convention to simplify your systems administration, and reduce errors caused by the wrong definitions being used. You need to consider several areas when deciding what names you should use.

Decide how many queue managers are needed and their names

Before you customise MQSeries you should consider how many queue managers you intend to have in your system, and develop an appropriate naming convention, for example, test MQSeries systems begin with T, and production begin with P, or MQT1 and MQP1. Other naming conventions include QxxP, QPxx, MPxx and MxxP for production regions.

A common problem is that two departments install MQSeries and each call their queue manager CSQ1, causing additional complexity when people tried to connect them together.

You might consider having a queue manager which reflects your company such as QMC1 where MC represents “My Company” or ABQA where AB is for the “Antarctic Bank.” This will make it easier to connect your MQ systems with MQ systems in other businesses. Queue managers on distributed platforms can have longer names. You may also need to coordinate your queue manager naming convention with other organisations your MQSeries system will communicate with.

You should have unique queue manager names within your network, including any cross enterprise networks. Having non unique queue managers makes the administration more complex and error prone.

You should plan for a queue manager for each of your environments, such as

1. Application development and test
2. Pre-production assurance where changes to the production system are run in a production like environment at maximum production throughput or higher. For example
    a. To ensure that all co-req changes are identified
    b. There are no unexpected changes to existing applications, such as no performance impact due to interaction with existing databases
    c. Any additional resources are available on the production system. This includes CPU, Database table space, Database logs, MQSeries page set and MQSeries logs.
3. Production

The limit to throughput of persistent messages is the rate at which data can be written to the log data set, so if you expect a high throughput you should plan for multiple queue managers. As a rule of thumb you can log a maximum of between 2MB and 4MB of data per second on RVA2 DASD, and between 6MB and 8MB of data per second on Enterprise Storage Server(Shark) DASD.

You also need to consider how you will test any application changes and any MQSeries service (PTFs). You might consider having one or more MQSeries test systems for this.

You should be able to support up to 9000 channels or clients on one queue manager when the message sizes are up to 30KB in size. With larger messages you will be able to support fewer channels. With 4MB messages you can support up to 350 channels. With 100MB messages introduced in version 2.1, APAR PQ33000, you can have less than 15 channels processing this size of message. This is because of the amount of virtual storage needed in the CHINIT address space. If you expect to use more than these number of channels you will need to use multiple queue managers.

The cost of starting and stopping channels is relatively high and depends on the number of channels which have been used (the depth of the SYSTEM.CHANNEL.SYNCQ queue). To minimise the CPU used by the CHINIT you should consider a second queue manager and CHINIT if you have more than about 2000 channels pairs which start and stop frequently.

If you are constrained for resources on your host, or expect a large number of channels then you might consider having a concentrator layer (three-tier instead of two-tier) This will offload a lot of the processing to the middle tier, so the host has to process only a few channels, and may achieve a high batch size

You should draw up a table of queue manager names, system name, TCP/IP address, LU name and description and ensure this is kept up to date. For example,

Queue Manager	**System**	**TCP/IP address (name)**	**TCP/IP port**	**LU name**	**Description**

Note: You should use names rather than numeric values for the TCP/IP addresses so in a disaster recovery scenario you have only to change a Name Server entry, rather than the channel CONNAME in every remote queue manager.

Decide naming convention for MQSeries executable library names

The name of your MQSeries executable libraries will normally be decided by the people who install the products on your system. Naming conventions like MQM.V210.SCSQAUTH or PP.MQ.V210.SCSQAUTH are typical. If the high level qualifier (for example, PP) already exists then you may not need any more RACF definitions to protect the libraries, but you will still need additional RACF changes for the queue manager.

You also need to consider how you will roll out a new release of a product. You should include the MQSeries product level as part of the MQSeries data set names so it is clear what release of the product is being used, but if you explicitly code this in jobs then these jobs will have to be changed when you change release. It is better to include the MQSeries libraries in a OS/390 JCL procedure and use the INCLUDE JCL statement to include them in jobs. Another approach is to have a symbolic in the data set name, and have a procedure to change the symbolic. With both of these ways you change one member in a procedure library and any new jobs submitted will pick up the changes.

Some customers have user a data set alias so the can change the name of the date set which the alias points to, and the user's JCL does not need to change. See Figure 1 on page 10 for an example.

```
//STEP1    EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     DEFINE ALIAS -
          (NAME(MQM.SCSQMACS) -
          RELATE(MQM.V210.SCSQMACS))
/*
```

Figure 1. Example of a Define Alias command

When you apply service to the system you may want to migrate queue managers one at a time to the new level of code. You could consider using different library names, for example,

- Installing service into MQM.V210.SCSQAUTH.INSTALL
- The production queue manages use MQM.V210.SCSQAUTH.PROD1
- Copy the updated libraries into MQM.V210.SCSQAUTH.PROD2
- Change the JCL for a Queue manager and CHINIT to use MQM.V210.SCSQAUTH.PROD2. Have the queue manager run for a time, perhaps a week, with these libraries to make sure that there are no problems with them.
- Repeat this for all of the queue managers and CHINITs.
- When all queue managers have been migrated to MQM.V210.SCSQAUTH.PROD2, then any new service can be copied into MQM.V210.SCSQAUTH.PROD1, and the cycle can be repeated.

Notes:

1. When copying libraries make sure that you copy all of the libraries, including load, ISPF panel, and source libraries.

2. When installing an emergency fix you need to remember that the system you install the fix on may be at a higher service level than the production systems. The approach of “just copy the affected module across” may not work as the production system may not have required pre-req fixes, and a modules may have one or more aliases which may be overlooked.

Decide naming convention for data sets required for each queue manager

You should keep the prefix for the queue manager data sets (eg MQMDATA) different from the MQSeries executable libraries, as the MQSeries product libraries should be read only to the queue manager and other MQSeries users.

A queue manager will require log and page set data sets. One naming convention is to have names like MQMDATA.MQP1.PSID0 where:

- MQMDATA can be used by Storage Management products to define how they are allocated, and when and how often the data sets are backed up.
- MQP1 is the name of a particular queue manager (4 letters)
- PSID0 is the name of the individual data sets.

You should define a storage group for MQMDATA so you can explicitly specify which volumes you want the data set placed on, (rather than letting the system choose for you) or can use non-SMS volumes. This is because logs and BSDS data set placement is critical for recovery in the event of the loss of one of these data sets. Check that the data sets are allocated where you specified.

The default jobs for allocating logs, BSDS and page sets, are suitable for a small test system and you should plan the size of your production data sets to match your production environment.

You could have your archive logs defined like MQMDATA.MQP1.ARCHA.*, and use Storage Management products to decide if data sets like MQMDATA.*.ARCHA.* get allocated to disk or to tape.

You might consider having data sets prefixes names like MQMDATA.MQP1.ARCATAPE and MQMDATA.MQP1.ARCADASD for duplexed archives, so that you maintain one archive on disk, and one on tape.

You also need to consider what names you call any backups, for example when you backup a page set you might call the backup MQMBACK.MQP1.PSID0.

You should have a separate library from the product libraries for the system parameter modules for example as MQMDATA.PARMS.LOAD. This is because the people who define the system parameter modules will need update access to this data set. This library will need to be APF authorised.

Decide naming convention for object definitions

As a general rule you should only use upper case object names, as using mixed case names tends to lead to confusion and error. If you will be using the console to process objects then only upper case can be used. The ISPF Operations and Controls panels and batch CSQUTIL jobs can use mixed case. Distributed queue managers often have object definitions in mixed case.

A good naming convention will simplify security definitions, as you can exploit generic definitions in your security manager. If you have a hierarchical naming convention then related objects will be displayed together.

You should keep the object names the same across development, test and production systems, as common definitions can be used, and this will reduce the chance of errors as changes are migrated into production. If you define or alter object definitions using a batch CSQUTIL job then this gives you better change control capability and you can be sure that the definitions that worked in test, work in production.

One approach is to have a member of a PDS for every update, and every week use a CSQUTIL job with the MAKEDEF option to offload all of the definitions to another member of the PDS. Any PDS updates before this can then be deleted (or possibly kept for audit purposes).

Decide naming convention for storage classes

A storage class based on the application is better than a name specifying the page set. For example STGCLASS(PAYROLL) is better than STGCLASS(PSID1) because you may move queues to different page sets and so STGCLASS(PSID1) may then map to page set 6.

Decide naming convention for IMS bridge storage classes

If you are using the IMS Bridge the storage class used for the bridge queue(s) must contain the XCF group and member name of the IMS subsystem. If only one IMS subsystem is being used, a name like IMSBRIDG or IBRIDGE is very clear. If you are using more than one IMS subsystem, consider using the IMS subsystem name in the storage class name.

Decide naming convention for queues

If you plan on having multiple queue managers, especially for workload balancing via clustering, then you should not have the queue manager name as part of the queue definition. You should consider having queue names like PAYROLL_SERVER and BRANCH_REPLYTOQ. If the dead letter queue is the same across all of the queue managers, then it will be easier to manage, and to set up systems administration products.

Decide naming convention for channels

You should decide what information you want as part of the channel name, for example origin and destination, and if you have multiple transport types (APPC and TCP/IP), the transport type. In a mixed protocol network some customers include the protocol as part of the channel name, but often this does not add much value especially when using long distributed queue manager names, and these names have to be abbreviated to include the protocol type. If you are using cluster channels you might call the channels TO.MQP1, if you are not using cluster channels then you might have a channel MQP1.BRANCH1. You could call your channel MQP1.TO.BRANCH1, but this may not work with a distributed queue manager, where the queue manager names can be longer than on OS/390.

You should make the transmission queue name the same as the remote queue manager name to simplify the remote queue definitions.

Decide naming convention for system parameter modules

A common naming convention is MQP1ZPRM and MQP1XPRM where MQP1 is the queue manager name, the ZPRM parameter is for the queue manager, and the XPRM is for the CHINIT.

If each queue manager has a different system parameter modules in a common library then you should consider deleting the default modules to prevent someone accidentally starting the queue manager without specifying the parameters.

You need to consider the advantages of having one load library per queue manager, or a shared library and differently named initialisation modules.

Decide where to archive logs

You need to decide whether you want to archive your logs to disk, to tape, or one to tape and the other to disk. If you archive to tape then when a tape is needed during recovery there is additional time needed whilst the tape is located and mounted. Reading backwards from tape is also slower than reading forwards. If you archive to disk then the archives may still be available on disk when needed, and not yet migrated. If the disk archives have been migrated then there will be an additional time while the migrated data sets are recalled before they can be used. You can use Storage Management products to decide if data sets like MQMDATA.*.ARCHA.* get allocated to disk or to tape.

You should periodically ensure that your archive logs can be recalled, and that there is enough space to recall the data sets.

Decide number, size and placement of log data sets

You need at least two active logs, preferably four or more, and you should have dual logging unless the DASD has built-in dual copy support. You should ensure that the dual copies are not on the same volume, or on the same physical device. You might consider having 4 volumes for each queue manager dedicated to active log data sets.

You can use the following OS/390 commands to help you decide the if the volumes you have chosen are suitable.

DEVSERV QDASD,VOL=volid	This displays the physical device a volume is on, so you can ensure that the duplex logs are on different physical devices.
D U,VOL=volid	This displays the device address for a particular volid
DEVSERV PATHS,devnum	This command allows you to check that Dasd Fast Write(DFW) is enabled for the device.

You should consider if you want to keep a whole day's worth of data in active logs, so if there is a problem archiving then the queue manager will be able to continue working.

An active log of 1000 cylinders of 3390 fit onto one 3480 tape, so this is a good size of log to consider.

You need to keep consecutive logs on different volumes to reduce disk contention. If you have an even number of active data sets in the ring, then you need at least two volumes, if you have an odd number then you will need three or more volumes. With dual logging this will be four volumes or six volumes. Whenever an active log fills up and wraps to the next log in the cycle, a checkpoint is taken. If the logs are too small then you get frequent checkpoints which uses additional CPU.

Deleting redundant archive log data sets

You need a procedure for deleting archive logs which are no longer needed, removing the entries from the BSDS and deleting the data sets. You need to keep archive logs for any page set backups that you wish to use, as these logs are needed to recover from a restored page set.

- You should issue the DIS THREAD command regularly (perhaps every hour) to ensure that there are no old units of work outstanding.
- As the start of a Unit of Recovery moves out of active logs onto archive logs, information about the thread is displayed on the console. This support is in PTF UQ33115, APAR PQ28088 on V120, and PTF UQ34749, APAR PQ2093 on V2.1.
- If you backup your page sets at least daily, and keep copies for two days, then you should be able to delete any logs older than three days. You will need all logs taken since the backup, and possibly up to three logs before the backup, but allowing for a complete extra day is easier to manage.

Decide the number and size of page sets

You must decide how many page sets you need. You need at least two, preferably three or more. One for object definitions, one for system queues (such as SYSTEM.CHANNEL.SYNCQ) and others for user messages.

If the applications which get messages from a queue are not working, you need to decide how long you want your applications to continue putting messages to the queue before getting queue full. You then need to allocate enough space in the page sets for these abnormal conditions. This applies to channels not processing messages and so transmission queues can fill up.

In a cluster environment where there is one common transmission queue per queue manager, you should consider the case where some or all of the channels are not started. There is a limit to the size of a page set (4GB on OS/390 V2.9) so you might want to put key queues, such as the cluster transmission queue in their own page sets.

You should keep short lived messages apart from long lived messages by defining the queues on different page sets and putting them in different buffer pools. One large page set is easier to administer than many smaller ones, but smaller page sets take less time to back up and restore, and I/O can be done to multiple page sets on different volumes, so checkpoints happen quicker. You should keep messages out of page set 0, this includes user messages, and system queues such as the Dead Letter Queue, or the SYSTEM.CLUSTER.TRANSMIT.QUEUE.

If you define secondary extents on the page set then the page sets can expand if it fills up.

Note: The task that causes page set expansion to occur is suspended until the expansion has complete. This could be seconds or many minutes depending on the primary and secondary allocations. If you cannot tolerate any delays to your applications then you should not use secondary extents and be sure to make your primary extent large enough.

You should not use full volume backups to backup your page sets if you have multiple extents. This is because the first page in the page set must be backed up before any other pages in the page set. If you have a multi extent page set on one volume the secondary extent may be backed up before the primary extent. If you have a multi volume page set then when the extents are restored there is a high risk of getting an inconsistent page set.

Mapping page sets to buffer pools

You should have buffer pool 0 only for page set 0, and separate out your long lived messages from the short live messages, see SupportPac MP16 (http://www.ibm.com/software/mqseries/txppacs/mp16.html)

Feedback to the author

The author would like feedback on how useful this book is so if you have found this chapter or the document useful please send an email saying so to Paice@uk.ibm.com. If you are willing to share your experiences on any topic in this document please send them, and the author will try to incorporate them in a future draft.

Implementing the queue manager

Define the priority of the MQSeries address spaces

You should make your production MQSeries the same, or similar priority to DB2. A typical order of dispatching priorities from high to low is JES, VTAM, TCP/IP, TSO, DB2, QMGR, CHINIT, CICS, BATCH.

If you are using goal mode, assign the Queue manager and the CHINIT to a service class with appropriate importance and goals. For example medium to high importance with a single goal using an intermediate execution velocity.

Decide a data set backup strategy

You can backup the MQSeries page sets when the queue manager is up (a so called fuzzy backup), or when the queue manager is down.

Backing up the page sets when the queue manager is down

The advantages of backing up the queue manager when it has been shut down is that the system can be restored from just the backup copies of the page sets themselves, and that active or archive log data sets prior to the backup are not needed. The log data sets since the backup will still be required.

You need to ensure that there are no outstanding units of work, such as indoubt transactions, when MQSeries is shut down. The DIS THREAD(*) command will show this.

You will need to keep any archive logs back to the oldest page set backup you have. When you decide that a backup is no longer required then you can delete the logs up to the next oldest backup. Make sure you keep the archive log that that matches the start of a backup.

If you wish to redefine the BSDS and log data sets then you must use the RESETPAGE command of CSQUTIL to reset the page sets.

Backing up the page sets when the queue manager is running

You can do a “fuzzy backup” of the page sets while the queue manager is running. When these backups are used, then the log data sets prior to the backup will be needed.

If you use a fuzzy backup then you will need all of the logs since the backup was taken, and possibly up to three logs before the backup was taken. In addition you need the logs for units of work which were active when the backup was taken. For short units of work, these will be in the logs just prior to the backup, but if you have long units of work then you will need more logs. You can use the DIS THREAD(*) command to display the START RBA of transactions, and can then determine which logs you will need by printing out the BSDS, or looking at the MQSeries console log for messages about ARCHIVE logs.

Create JCL to backup up page sets

You can backup your page sets while the queue manager is running.

On a busy MQSeries system using persistent messages you should backup your page sets twice a day or more frequently. With modern DASD which has dual copy or similar facilities once a day may be acceptable. However you still need to be able to recover from situations like someone accidentally deleting the data set.

Using DFDSS is faster than using VSAM repro, but using Snapshot on RAMAC Virtual Array or FlashCopy on IBM Enterprise Storage Server, a backup or restore can be done in seconds regardless of the page set size by exploiting the facilities of the DASD, see Figure 2 on page 16 for an example of how to do this.

```
//COPY1     EXEC PGM=ADRDSSU,REGION=32M
//SYSPRINT DD SYSOUT=H
//DISK1    DD   DSN=MQMBACK.MQP1.PSID0(+1),
//            DISP=(NEW,CATLG),
//            SPACE=(CYL,(2000,500),RLSE)
//SYSIN    DD *
 DUMP -
        DATASET(INCLUDE(MQMDATA.MQP1.PSID0)) -
        OUTDDNAME(DISK1) -
        TOL(ENQF)  -
        CC   -
        SPHERE -
        ALLDATA(*) -
        ALLEXCP -
        CANCELERROR
/*
```

Figure 2. Sample JCL to backup a page set using DFDSS

If you are using HSM to backup the page sets check that HSM will backup the page sets if they are in-use, by setting the SMS Management class parameter Backup Copy Technique to “Prefer concurrent.” If you are using DFDSS then you will need the TOL(ENQF) definition as shown above. Check that backups are being taken, and that they can be restored on the running system, and any remote backup system.

You can use the TSO commands in Figure 3 to display the status of HSM backups

```
HLIST LEVEL(MQMDATA.MQP1) BCDS
HLIST DSN(MQMDATA.MQP1.PSID0) BCDS
```

Figure 3. Sample TSO commands for displays the status of HSM backups.

Consider taking backups into a generation data group(GDG) so you have at least two backups. Make sure you do not delete the last backup as part of the backup job in case you lose the data set during the backup and you will then not have a valid backup. Figure 4 shows a example of how to do this.

```
//STEP1    EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=*
//SYSIN    DD     *
    DEFINE GENERATIONDATAGROUP -
           (NAME(MQMBACK.MQP1.PSID0) -
            SCRATCH -
            LIMIT(5))
/*
```

Figure 4. Sample JCL to define a GDG for a page set backup. This JCL defines 5 data sets to be used for backup of the page set. A reference to MQMBACK.MQP1.PSID0 will use the most recent version.

Allocate data sets to contain JCL for a queue manager

You should consider having a data set to contain the JCL for each queue manager. So for a queue manager MQP1, you might have MQMDATA.MQP1.JCL. In this data set you have all the jobs required for that queue manager, for example:

1. To define, assemble and linkedit the system parameter modules
2. A job for each page set to delete the page set and define it.
3. Jobs to back up page sets and a job for each page set to restore it from a backup. You need one job per page set as you usually need to restore one page set at a time and not all of them
4. Jobs to list the log data sets in the BSDS
5. Process MQSeries statistics (see SupportPac MP15 in “Monitoring your systems”)

Monitoring your systems

You should collect MQSeries statistics from your system on an hourly or half hourly basis. You should collect SMF record types 115 and 116 for offline analysis. SupportPac MP15 is a program to print out the statistics.

You should monitor the following:

1. For each buffer pool QPSTSOS,QPSTSTLA,QPSTDMC should always be zero and QPSTDWT and QPSTRIO should typically be zero. If these are not zero then make the buffer pool bigger in your CSQINP1 data set.
2. For the log manager QJSTWTB should always be zero - if not then increase OUTBUFF in CSQ6LOGP and re-assemble it

You should monitor the statistics on a weekly basis and do trend analysis on buffer pool usage.

The cost of collecting statistics is small, of the order of a few percentage overhead in the queue manager.

If you are already using systems management and/or monitoring tools for other OS/390 systems (for example, Tivoli) they may have an MQSeries component that you can use.

As well as using MQSeries statistics you should also monitor queues such as event queues and your key application queues. You can have events produced if the queue depth exceeds specified values. If these events are produced this may indicate an application or operating problem. Systems management products usually provide processes for monitoring and reporting on the queue depths and events.

You should monitor transmission queues, Dead Letter Queue, and MQSeries Integrator(MQSI) and MQSeries Workflow queues in addition to application queues. You can also use the performance events to report out of line response times.

Do you need accounting information?

Accounting data is reported when the task or channel ends. So if you have a channel active for a week, then one record will be produced at the end of the week. This is different from statistics that are collected on a time interval. The overhead of using the accounting trace is approximately about 10% in the worst case of a CICS application putting and getting a short 1 KB non persistent message. For larger messages or more messages per transaction, the percentage overhead is less.

You might use accounting information to charge back to the end users. You can do this by CPU used or by number of messages processed.

If you do not plan on using the accounting data then you should turn accounting trace off.

Decide on a security strategy

You need to have a system wide security strategy for your MQSeries systems, which includes access to and from MQSeries systems outside of your organisation. Once you have decided on your strategy you can work with your security people to decide what level of security you need on MVS. For example, you need to consider how secure the remote queue managers are, and how much you can trust userid information in the messages.

You should have controls in place to control who can define, alter or delete queues.

You need an external security manager, such as RACF, to control who has access to messages on queues and who can issue MQSeries commands. For CICS regions, if you have security to control who can access transactions, this may be enough, and you may decide not to restrict access to queues. If your applications have some built in security, additional security for opening queues or or putting and getting messages from queues may not be needed. You may still need to control who can create alter or delete objects.

Many customers have the approach that if a user is allowed to use a CICS transaction then they do not need to have queue level security. If you are considering this then you need to be confident that people do not have unauthorised access to the CICS regions, for example:

1. Production CICS regions must be authorised to connect to the production MQSeries
2. Test CICS regions are not authorised to connect to the production MQSeries.
3. You have CICS transaction security to restrict who can access particular transactions
4. Transactions and programs cannot be defined by unauthorised people
5. Only approved changes to application are implemented on the production regions.
6. Test people do not have access to production CICS regions.

You should decide if you need to set up groups of userids to be able to access queues. A good naming convention and the use of generic or group definitions in the security manager can simplify the security definitions. MQSeries allows group definitions like MQMADMIN and GMQADMIN.

You need to decide if you use the userid of the task putting the message, or use the userid in the messages (the alternate userid).

When messages are put to queues from remote queue managers the channel may not be authorised to put messages to the queue, and the messages will be put on the Dead Letter Queue (DLQ). You should be prepared to process messages on the DLQ and decide what actions you want to take for any messages which are there for the not-authorised reason.

MQSeries can cache security information within the queue manager. If you decide that every MQOPEN request will be logged by the security manger then the Queue Manager will not cache the information and there is a significant overhead in doing this.

If you have many thousands of userids which connect perhaps just once a week for a short period, you might consider reducing the security time interval to reduce the amount of security data held in the MQSeries cache. See Figure 5 on page 19 for an example command. This can give a small performance benefit, and reduce the amount of virtual storage needed by the queue manager.

```
ALTER SECURITY INTERVAL(10) TIMEOUT(0)
```

Figure 5. Alter security command for any userids with low usage

Decide if you want to use MQSeries events

You need to decide the best way of detecting problems, for example your business logic may report problem as part of your standard error reporting, and so MQSeries events may not be required. If you are using events, you need to have a program or monitor to process the event queues. You should decide on an enterprise wide basis if you want the events to be handled by a central tool, or to be handled on each platform. For example, if you manage events centrally, you may not get notified of a channel stopped, as the event cannot be sent if the channel has stopped.

Vendor products often use the event queue for displaying and reporting events. You need to check that you are collecting the right information, for example, queue high as well as queue full. If you only collect the queue full event you will not get notification that a queue is getting full.

What level of availability do you need?

You need to review what level of availability you need. This covers applications as well as the MQSeries system. For example, do you have single points of failure on your systems. If you use MQSeries clustering then if you have multiple server queue managers you can shut one down and the client queue managers will continue to run with no additional systems operations.

Do you have applications which could be a bottleneck, for example, do all messages have to pass through one queue or server?

You should separate out application availability issues from “infrastructure” availability issues (for example, channels) as the solutions are different. For example, applications need to be able to handle queue full conditions, messages not arriving in time, and messages arriving in a different order if a backup channel is used.

You need to decide how other address spaces required for your applications will be started, such as the rules engines for MQSeries Integrator.

You should consider using the OS/390 ARM facilities to restart the queue manager and CHINIT after a failure.

Open edition setup

Figure 6 shows some sample definitions for defining a userid to be used for the queue manager and the CHINIT using Open Edition sockets support

```
ADDUSER MQTASK SUPGROUP(SYS1) OWNER(SYS1) (1)
ALTUSER MQTASK OMVS(UID(n))               (2)
ALTGROUP SYS1 OMVS(GID(m))                (3)
RDEFINE STARTED MQP1MSTR.* STDATA(USER(MQTASK)) (4)
RDEFINE STARTED MQP1CHIN.* STDATA(USER(MQTASK)) (5)
SETR RACLIST(STARTED) REFRESH             (6)
```

Figure 6. Sample MQSeries started task RACF definitions

The numbers in brackets are as follows:

1. Define the userid called MQTASK to RACF

2. Give the MQTASK userid an OMVS segment to allow it to access Open Edition facilities. “n” has to be a numeric value, on the IBM MQSeries test systems this is “1.”
3. Defines the MQTASK userid as being part of the Group id. “m” is numeric and on the IBM MQSeries test systems this is “1.”
4. Define the started task name MQP1MSTR to use userid MQTASK
5. Define the started task name MQP1CHIN to use userid MQTASK
6. Refresh the started task definitions so they can be used on this system

If the started task userid is defined as being able to logon to TSO, then you can check that Open Edition is set up properly by checking that the OPING command to a remote system works.

CHINIT and channel setup

Channel parameters

As a general rule, use the defaults where possible.

You need to consider both ends of the channels when setting channel parameters, as many parameters are negotiated.

Using cluster channels greatly simplifies the administration needed to define channels.

It is relatively expensive starting and stopping channels, so you should review what disconnect time you need. You need to have a disconnect interval long enough to “cover” the small gaps between messages arriving on the xmit queue, but not over long, in that no messages are processed for a long time. For example, if you have a branch network where a branch sends a group of messages to the server perhaps once a day then a disconnect interval of 10 minutes may be acceptable. If you have a hundred messages a day being processed at random times, then having the channel start when the first message is processed, and have it time out after two hours might be a acceptable. A shorter DISCINT value is better able to tolerate an unreliable network.

Although you can get a small saving if you specify a batch size of one, and you have a low message rate, you should generally use the default batch size of 50 because you might need a larger value if the message rate increases; for example, if you are running in production rather than in test or more applications are brought online.

A batch size of 50 is good for short (less than 5KB) messages, but you should use a smaller batch size for larger messages, with a batch size of one for 1MB messages.

Heartbeat versus TCP/IP Keepalive

On V1.2 and later releases of MQSeries channels support heartbeats. Internal data is sent down to the remote end, and a reply sent back. If there is problem with the link then this will be detected and the channel will stop and try to restart. This gives you the opportunity to resolve problems even when there are no messages to send. The heartbeat exchange gives the receiving queue manager the opportunity to quiesce the channel.

TCP/IP has a facility called “Keepalive” which provides the same sort of facility for all of its connections. You should use heart beats in preference to keep alive as it can be specified on individual channels and the heartbeat interval can be set appropriately for each channel. You specify TCPKEEP=YES in the CSQ6CHIP definitions to use this facility.

If you have a low disconnect interval, for example 5 minutes, then heartbeats may not be necessary. If you have a long disconnect interval then a heartbeat interval of 5 minutes may be satisfactory.

The heartbeat interval, if used, should be less than the disconnect interval. If the heartbeat interval is too low then this can cause significant additional network traffic. The optimum value depends on the reliability of the network and the availability of remote queue managers. If the heartbeat interval is too short then there will be additional network traffic, and additional CPU will be used. If the interval is too long, you might not detect network problems in the optimum time.

Check the CHINIT has the correct dispatching priority (high)

You need to consider how your CHINIT will be started, for example you can set up your QMGR procedure to automatically start the CHINIT and have it start the listener. You should also consider how you will restart the CHINIT and Listener if there is a major communications failure.

You should also have documented procedures on what to do if a channel fails to start, for example:

- Ping the remote system
- Check the remote CHINIT is available
- Check the Inet daemon is running on the remote system, if applicable.

If a channel does not start then check the messages at both ends.

To check that your CHINIT is set up properly you can do the following test using the following unusual setup.

1. Define a sender channel which points back to itself, see Figure 7 for an example.

```
DEF CHL(SELFTEST) CHLTYPE(SDR) CONNAME('MYSYSTEM(1414)') XMITQ(TEST)
DEF QL(TEST) USAGE(XMITQ)
```

Figure 7. Define objects for self test

   You would not normally define a channel back to itself, but we want the channel to fail to start.

2. Start the channel. This will fail with the messages in Figure 8.

```
+MQP1 STA CHL(SELFTEST)
CSQM134I +MQP1 CSQMSCHL  STA CHL(SELFTEST) COMMAND ACCEPTED
+CSQX500I +MQP1 CSQXRCTL Channel SELFTEST started
+CSQX502E +MQP1 CSQXRESP Action not allowed for SELFTEST
+CSQX599E +MQP1 CSQXRESP Channel SELFTEST ended abnormally
CSQX547E +MQP1 CSQXRCTL Remote channel SELFTEST has the wrong type
CSQ9023E +MQP1 CSQXCRPS ' STA CHL' ABNORMAL COMPLETION
+CSQX547E +MQP1 CSQXRCTL Remote channel SELFTEST has the wrong type
+CSQX599E +MQP1 CSQXRCTL Channel SELFTEST ended abnormally
```

Figure 8. Example messages when sender channel is defined as a loopback. Because the channel was defined to point back to the originating queue manager it will fail. (The command prefix string is +MQP1.)

   If the Listener is not available, you get message “CSQX202E +MQP1 CSQXRCTL Connection or remote listener unavailable”

TCP/IP set up

If you are using OE sockets then make sure your name server is set up properly, for example you should be able to use the Open Edition command OPING to the remote computer. If you have problems check that NSINTERADDR is specified correctly either in the /etc/resolv.conf file by using the TSO command OEDIT, or, if file does not exist, in the data set in the //SYSTCPD DD statement. Restart the channel when the OPING command is successful. Figure 9 on page 23 is an example file.

```
TCPIPJOBNAME TCPIP
IBMMVS25:    HOSTNAME  IBMMVS25
DOMAINORIGIN  HURSLEY.IBM.COM
NSINTERADDR   9.20.4.6
RESOLVEVIA UDP
RESOLVERTIMEOUT 5
RESOLVERUDPRETRIES 1
DATASETPREFIX PP.TCPIP.OS39025
```

Figure 9. Sample /etc/resolv.conf file for OE sockets interface

If you are not using the /etc/resolv.conf file then specify the TCP/IP data set in the //SYSTCPD DD JCL statement. If this statement is not present then the data set will be dynamically allocated whenever a channel is started which will increase a channel's start time.

Ensure you have a large MAXPROCUSERS. You can check your values using the D OMVS,O command, see Figure 10 for example output.

```
MAXPROCSYS      =        500    MAXPROCUSER     =        512
MAXFILEPROC     =      65535    MAXFILESIZE     = NOLIMIT
MAXCPUTIME      = 2147483647    MAXUIDS         =        500
MAXRTYS         =        256    MAXPTYS         =        256
MAXMMAPAREA     =       4096    MAXASSIZE       = 2147483647
MAXTHREADS      =       1024    MAXTHREADTASKS  =       1024
MAXCORESIZE     =    4194304    MAXSHAREPAGES   =      32768
IPCMSGQBYTES    =     262144    IPCMSGQMNUM     =      10000
IPCMSGNIDS      =        500    IPCSEMNIDS      =       4096
IPCSEMNOPS      =      32767    IPCSEMNSEMS     =         25
IPCSHMMPAGES    =      25600    IPCSHMNIDS      =        500
IPCSHMNSEGS     =       1000    IPCSHMSPAGES    =     786432
SUPERUSER       = BPXROOT       FORKCOPY        = COPY
STEPLIBLIST     = /etc/steplib
USERIDALIASTABLE=
PRIORITYPG VALUES: NONE
PRIORITYGOAL VALUES: NONE
MAXQUEUEDSIGS   =       1000    SHRLIBRGNSIZE   =   67108864
SHRLIBMAXPAGES  =       4096    VERSION         = /
SYSCALL COUNTS  = NO            TTYGROUP        = TTY
SYSPLEX         = NO
```

Figure 10. Sample output from D OMVS,O

APPC set up

If you are using APPC check the log modes you are using as you can get significant performance improvements using a higher RU size and pacing.

Other channel considerations

If you want to encrypt the flows, how will you ensure that the keys are protected?

It may be worth doing compression to reduce the amount of traffic flowing between queue managers, but with an increased CPU cost in the queue managers.

How will you perform security checking when the channel starts?

With MQSeries clients how do you distribute the MQSeries client code?

How do you distribute a Client Definition File?

Are the right code pages being used?

Planning for applications

Before you define queues for applications, you need to obtain the following information from the application programmers

1. Message persistence. Your application should explicitly specify the persistence as other techniques, such a queue default persistence, seem to cause problems
2. Maximum message size
3. Average message size
4. Maximum number of messages expected on the queue. This should include the cases when a channel to a remote queue manager is not working, or the application processing a queue is not available. You need to consider how long you will allow messages to be put before the queue fills up.

   From the average message size and the maximum number of messages expected on each queue you can estimate how much space you need for page sets.
5. Are the messages expected to be processed shortly after they are created or are they expected to remain for half an hour or more? This will affect the buffer pool which is used. See “Decide the number and size of page sets” on page 13 and SupportPac MP16 for more information.
6. Will any applications be using MQSET to change queue attributes?

   It should be clearly explained and documented why this is needed, and what happens if the application fails to reset an attribute.
7. What happens to messages which the application cannot handle? Will the application have its own “Dead Letter Queue” or use the queue manager's DLQ? The applications will have to explicitly put the message to the chosen dead letter queue, with the appropriate header as required.
8. If an application fails, or issues a rollback command, then the backout count of the message is incremented. You can use a facility called mark skip backout to prevent an endless application loop where the application gets a message, it cannot process it, so it rolls back and gets the same message again and so on. You can use the backout threshold(BOTHRESH) queue attribute so your application can put the message it is unable to process onto the backout queue name(BOQNAME).
9. Are alternate user or context fields set by the application which will affect the security checking of messages?
10. What events will be generated from this queue, and what will process any events produced?
11. Will triggering be used for these queues? If so are the associated queues defined?
12. Only use trigger type First, as using other trigger types tends to lead to confusion and badly designed applications.
13. Will access to the queues be controlled by an external security manager?
14. If dynamic queues are used, what is the process for emptying and deleting any queues that are not cleaned up by the application

15. If a common queue is used for replies, how will you delete messages which arrived after an MQGET timed out; so called orphan messages? This could be done using expiry, or processing the queue overnight when the normal applications are not running.
16. If the queue is a remote queue, are the appropriate channel definitions available?
17. If persistent messages are used, then this will impact the amount of DASD used for logging, and so more archive logs may be created. You might need to add more active log datasets to the queue manager to be able to keep a day's worth of log data in the active logs.
18. You should generally have a small number of messages in a unit of work, with fewer messages the larger the message are. For example, up to 50, 1 000 byte messages, and one 100 000 byte message per unit of work.
19. Ensure that units of work do not exist for a significant time. For example, do not get a message within syncpoint and then prompt the user for a decision.

Application design question

You should review the following questions related to application design as part of the design and as part of the testing.

If you are using messages of different priorities, or clustering where messages can get routed through different paths then it is possible for messages to arrive out of sequence, so, for example, a high priority request to cancel an order may arrive before the order itself. Your applications need to be able to handle this sort of message arrival pattern.

What do applications do if they get errors like queue disabled, or queue full? Do they report an error or wait for a short period and retry?

For the first request in a unit of work, you should specify fail if the queue manger is quiescing. If your transaction has multiple MQSeries requests, you should consider if you want the transaction to proceed, or to backout the transaction if it gets notified that the queue manager is quiescing.

If the channel, transaction or job which process messages on a queue is not working, then messages may build up on the queue. You need to decide what the maximum queue depth you are prepared to accept before having the putting applications fail with queue full.

What happens to messages on the dead letter queue, do you have a monitor for this?

Is there appropriate triggering on queues? Generally it is more efficient for the triggered application to get and process each message in turn rather than be triggered for each message.

In general, applications should not use MQSET to change queue attributes. If applications use set commands to alter queue attributes what happens if the application fails, for example, after setting a queue to no-trigger. The applications should have logic to reset the change, for example, change the queue back to trigger regardless of how the application ended, whether the application abended, was cancelled or ended normally. This may be difficult to do.

You need to have a policy on using mark skip backout.

Applications should check the length of messages and data. You need an application philosophy if a wrong length message or an invalid piece of data is detected

You should identify the most significant bottleneck in the application, for example a process which single threads messages. If this process can not have multiple applications processing a queue then this is likely to be a bottleneck.

You need to instrument your application so you can tell when you are approaching a limit. For example:

- Using CICS you can get CICS transaction response times recorded in RMF every half hour.
- Monitoring the queue depth, and producing an event if the queue depth gets above a certain percentage full.
- Have the application record the time for key calls, such as to MQSeries or DB2 and write an event to an application specific queue if the response time is out of specification.

If the response time increases significantly with workload then you are likely to be approaching a limit. This could be due to insufficient CPU, increased paging, I/O contention, Database locking or simply poor tuning.

Several customers have exploited MQSeries by using different queues for different criteria, for example account names beginning with A-H go to one queue, I-S to another, and T-Z to a third queue. One CICS region processes messages on one queue, and by having the DB2 database partitioned A-H, I-S and T-Z, the CICS regions update DB2 and avoid database contention.

What should the application do when there are authorisation failures?

What do you plan to do if you lose a queue manager data set?

You need to consider what recovery actions you will take in the event of a loss of an MQSeries data set. For example, you should be backing up the page sets at least twice a day, using DFDSS BACKUP, even if the page sets are not used. What is the maximum restart time you can tolerate?

Schedule and perform recovery tests

You need to be prepared to recover from a failure in any component.

For MQSeries this means that you have to be able to recover from a damaged page set or damaged log, as well as failures in other components.

Other administration tasks

How will you delete any dynamic queues which have not been deleted automatically?

What testing do you need to do?

You need to be prepared to test different sorts of changes to your system.

1. A new release of MQSeries
2. PTFs for MQSeries
3. New applications
4. Changes to existing applications
5. Definition changes, such as changing queue attributes
6. “Systems programmer” type changes such as turning accounting or security checking on or off.

Tasks that should be done on a regular basis

The list of tasks below should be done periodically, typically weekly.

1. Check MQSeries statistics
2. Monitor queue depths
3. Monitor resources used by MQSeries
   - CPU
   - virtual storage
   - DASD usage
   - Buffer pool utilisation
   - Page set utilisation
4. Check for security violations
5. Check that the MQSeries data sets are optimally placed, in case they were moved to other volumes, for example, they were moved to balance the I/O to DASD.
6. You should periodically ensure that your archive logs can be recalled, and that there is enough disk space to recall the data sets.

Operational tasks

Startup messages when long unit of work are present

At restart, if long inflight units of work are detected you are prompted if you wish to commit the unit of work, or let the system backout the updates. You need to have a procedure for handling this. If you decide to commit every long unit of work then the restart will be faster as archive logs will not be needed as part of the backward recovery stage of restart. You need to decide if you will commit the unit of work, and what the impact will be on your data if you do so. This function was introduced in the following PTFs, UQ31765 on V120 and UQ34577 on V2.1. See SupportPac MP16 for an example of this function.

Automated processing of MQSeries messages

Is there automation to detect MQSeries error messages, i.e. all CSQ*E messages.

Review new messages for inclusion in automation

New messages may be introduced in a new release of the product, or through the service channel. You should review these and decide which of these you can suppress, and which you need to take actions on.

What should Operations staff monitor?

You need to decide what the Operations staff should monitor and how to detect an unusual condition or problem. This might include issuing commands like DISPLAY DQM to make sure the CHINIT is working and the listener is started.

Develop operational procedures

The Operations staff need documented procedures to cover the tasks they are likely to have to handle.

Some examples include:

- How to start and stop the queue manager
- How to respond to startup messages see “Startup messages when long unit of work are present” on page 27
- How to start and stop the CHINIT
- How to start and stop the listener
- How to start and stop channels
- How to start MQ related applications such as MQSeries Integrator or MQSeries Workflow
- What to do if there are messages about problems with the active or archive logs
- Who to contact if there are situations which they cannot handle
- How to request additional assistance

Sending your comments to IBM

MD16: MQSeries for OS/390 -
Getting the best from MQSeries for OS/390

MD16 SCRIPT

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book. Please limit your comments to the information in this book and the way in which the information is presented.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

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

You can send your comments to IBM in any of the following ways:

- By mail, use the Readers' Comment Form.
- By fax:
  - From outside the U.K., after your international access code use 44 1962 841409
  - From within the U.K., use 01962 841409
- Electronically, use the appropriate network ID:
  - IBMLink: IBMGB(AIMPACS)
  - Internet: aimpacs@uk.ibm.com

Whichever you use, ensure that you include:

- The publication number and title
- The page number or topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.

Readers' Comments

MD16: MQSeries for OS/390 -
Getting the best from MQSeries for OS/390

MD16 SCRIPT

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. This form is provided for comments about the information in this manual and the way it is presented.

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

Be sure to print your name and address below if you would like a reply.

Name	Address
Company or Organization	
Telephone	Email

A guide for users of MQSeries for OS/390
MD16 SCRIPT

1 Cut along this line

You can send your comments POST FREE on this form from any one of these countries:

Australia	Finland	Iceland	Netherlands	Singapore	United States of America
Belgium	France	Israel	New Zealand	Spain	
Bermuda	Germany	Italy	Norway	Sweden	
Cyprus	Greece	Luxembourg	Portugal	Switzerland	
Denmark	Hong Kong	Monaco	Republic of Ireland	United Arab Emirates	

If your country is not listed here, your local IBM representative will be pleased to forward your comments to us. Or you can pay the postage and send the form direct to IBM (this includes mailing in the U.K.).

2 Fold along this line

By air mail
Par avion

IBRS/CCRI NUMBER: PHQ - D/1348/SO

NE PAS AFFRANCHIR

NO STAMP REQUIRED

IBM

REPONSE PAYEE
GRANDE-BRETAGNE

IBM United Kingdom Laboratories Limited
Information Development Department (MP 095)
Hursley Park
WINCHESTER, Hants
SO21 2ZZ United Kingdom

3 Fold along this line

From: Name ______________________
Company or Organization ______________________
Address ______________________
____
EMAIL ______________________
Telephone ______________________

1 Cut along this line

4 Fasten here with adhesive tape