MQEdit

Live Parsing Editor for IBM MQ Messages
User Guide

Version 9.3.3
18th March 2024

Paul Clarke

MQGem Software Limited
support@mqgem.com

Take Note!

Before using this User's Guide and the product it supports, be sure to read the general information under "Notices".

Twenty Fourth Edition, March 2024

This edition applies to Version 9.3.3 of *Live Parsing Editor for IBM MQ Messages* and to all subsequent releases and modifications until otherwise indicated in new editions.

Notices

The following paragraph does not apply in any country where such provisions are inconsistent with local law.

The information contained in this document has not been submitted to any formal test and is distributed AS IS. The use of the information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item has been reviewed by MQGem Software for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

IBM MQ
IBM
AIX
OS/400
MVS
z/OS

The following terms are trademarks of the Microsoft Corporation in the United States and/or other countries:

Windows 95, 98, Me
Windows NT, 2000, XP

Table of Contents

Figure

Introduction

Here at MQGem we frequently get requests for new products. By far the most common request has been for a Message Editor. Well, we have finally got around to developing one and we hope you are as pleased with the results as we are.

It would have been easy for us to have just developed a simple hex editor. However, MQ messages tend to be more complicated than just a string of bytes. They have structure. What we wanted was a product which would allow you to easily edit the message and have it maintain that structure. Indeed, even allow you to modify the structure in a simple fashion if you wish.

Of course we realise that we have only touched the surface of what could be done in an MQ Editor. We fully expect there to be many follow-on releases driven by the needs and wants of our customers. However, hopefully you will find that the features already in the editor make modifying MQ messages much simpler, quicker and more accurate making this a product useful for Administrators, Testers and Developers alike.

My thanks go to Morag Hughson for considerable assistance with this manual, for numerous usability suggestions, for keeping me sane when nothing ever seems to go right, as well as tirelessly testing many buggy versions of the program!

If you have any questions about the product or perhaps suggestions for additional features then please send an email to support@mqgem.com. Equally if you don't have a licence and would like to try out the editor then please send an email to the same address.

Main changes from previous version

Unless otherwise stated the behaviour of the previous version should be maintained and, if all goes well, enhanced. While every effort is made to try and ensure that there are as few bugs as possible it would be surprising if some problems didn’t leak out. We would recommend that users keep their previous version of MQEdit handy so that they can revert to it if a bug is found. Needless to say, please report any incompatibilities to us if they are found and we will do our best to provide a fix.

The following changes have been made in this version:

1. **Support for Over-punched and separate signed display fields**
   Support for this field type makes display and editing of COBOL copybooks much easier. Please see Over-punched Signed Decimal Fields and Signed Decimal Character Fields on page 78 for more information.

2. **New actions in location dialog**
   You can now 'Connect' and 'Disconnect' and view your error log and INI files direct from the location dialog.

3. **Date and Time Fields**
   MQEdit now allows you to display Date and Time fields in a single fields. This makes the message list easier to sort and allows the user to choose their own date format. For more information please see 26.1. Dates and Times on page 130.

4. **Progress Dialogs**
   MQEdit will now report on its progress as it is loading, unloading, moving, copying or deleting large numbers of messages. For more information please refer to section 17.3. Progress Dialog on page 97.

5. **Import from MQ Explorer export file**
   It is now possible to take the XML file generated from an MQ Explorer export and import the location definitions in to MQEdit. This makes it quicker to get started in MQEdit if you have previously been using MQ Explorer. You can set the Network Names and Groups to the values you want or take the information about MQ Explorer sets for this purpose. For more information please see From MQ Explorer File... on page 99.

6. **Import from JSON CCDT file**
   Enhancing the pre-existing CCDT import dialog, JSON CCDTs can now be used as an import source for queue manager connection information, just like Binary CCDTs.

7. **Ability to specify whether the location is 'MVS' when importing from a CCDT**
   This applies to both the standard binary CCDT and the JSON CCDT.

8. **Import now allows you to specify Network Name values**
   All queue manager import dialogs now allow you to set the Network Name values at import time.

9. **Column Filters**
   This release adds column filters to the message list. MQEdit has always had very flexible filtering using the message list filter field. However, this can appear complicated to the novice user. Column filters provide a quick and simple way of filtering by the value in a particular column of a list. For more information please see 15.1. Column Filters on page 54.

10. **Formatted Put Date and Time**
    When viewing the Message Descriptor in the Message Edit Pane, the Put Date and Put Time fields will show a formatted version over on the right, just like user format date fields, to ensure the date conveyed is unambiguous.

Chapter 1. Live Parsing Editor for IBM MQ Messages

This document describes the functions available in the program.

1.1. Overview

The main window displays a list of Queue Managers. Typically, this list would contain any local Queue Managers and a number of other 'remote' Queue Managers that the machine has access to via an MQ client connection.

The user can then perform the following:

- **Browse the Queues**
  Each local queue can be browsed showing a brief summary of the contents of the messages. Which fields of the Message Descriptor should be shown is selectable.

- **Browse a message**
  An individual message can be shown in full, either formatted or in hex.

- **Edit a message**
  The contents of the message can be edited and the message replaced on the queue. The editor provides a range of features:

    - Full live parsing and display of message content
    - Editing either in formatted mode or in hex
    - Display different levels of message detail
    - Message editing and display in a wide range of ASCII and EBCDIC codepages
    - Display hex bytes in predictive text (for example Correl Id content)
    - Full undo/redo support
    - Find and Replace capability
    - Full copy/paste support
    - Bookmark and restore message position
    - Add and remove message parts, for example build up PCF messages effortlessly
    - XML assistance such automatically updating the partner tag

- **User Format Messages**
  You can define the structure of your own messages and associate the structure with a particular Queue or Format.

- **Move, Copy or Delete Messages**
  Messages can be copied or moved either in bulk or individually by selection or drag-and-drop. If required messages can also be deleted.

1.2. Installation

Create a directory, say MQEdit, and copy the installation zip file into it. You now need to unzip this file; you should be able to use any of the regular zip utilities to do it. For example, in Windows Explorer you should be able to right click on the file and select 'Extract All..'.

Once you have unzipped the file you should end up with a directory containing the following:

File	Description
mqedit.exe	The actual editor program itself
mqeditstart.cmd	A simple command file for starting the program that you can choose to use if you wish
mqedit.pdf	The MQEdit manual....probably what you are now reading
readme.mqedit	A simple text file describing the change history of MQEdit

mqedit.exe is the main executable of the MQEdit program. Note that the directory where you place the .EXE program is the default location where the program will store its configuration file MQEDIT.CFG and backup file MQEDIT.BAK.

You generally need not worry about these files but you should be aware that they are written and therefore the program needs write access to the directory. Alternatively you can use the '-f' parameter to the program to give the directory name the program should use for these files.

The MQEdit program will dynamically load either the MQ client or MQ Queue Manager libraries as required. So, in order to successfully connect to a Queue Manager at least one of these IBM MQ products must be installed. The windows IBM MQ client is available for free download at:-

http://www-01.ibm.com/support/docview.wss?uid=swg24032744[1]

1.3. MQEdit Versioning

Before we start it is probably worth saying a few words about MQEdit versioning. The version of MQEdit and the version of IBM MQ need not be the same but they are related. Essentially the first two digits of the MQEdit version tells you the level of MQ that MQEdit understands. When IBM brings out a new version of MQ, let's say MQ 10.1, then MQEdit 9.0 will almost undoubtedly run on that version but it will not know about any new message formats or reason codes. If MQEdit detects a Queue Manager which is at a later version than itself it will present a warning dialog. You may choose to ignore this dialog but be aware that some messages may not be formatted as completely as they could be.

For this reason it is always advisable to run a version of MQEdit where the first two digits are equal to or greater than the version of MQ you are running. So, to reiterate, you can run any version of MQEdit on virtually any version of IBM MQ. Of course the later the version of MQEdit the more features you will have. It is perfectly reasonable to run MQEdit 9.0 against MQ 5.3 or 6.0. The only thing you should try and ensure is that the version of MQEdit you are using is late enough to understand all the MQ attributes. And the simplest way to check that is to check the first two digits of the version.

---

[1] Note that there may be a more recent version of the MQ client (or indeed an earlier version) available on the MQ SupportPac web sites. It is recommended to use the latest version possible, however, MQEdit should work with any version.

1.4. Running on Linux

Although MQEdit is a Windows program it is possible to run it on Linux using WINE. WINE is a fabulous piece of free software that comes from the WINE project (https://www.winehq.org/). This software allows a huge range of Windows applications to run, virtually unchanged, on an Intel Linux machine[2]. The software is not an emulator but rather translates Windows API calls to POSIX calls on the fly which means that it is fast and efficient. The level of compatibility is very good and it is well supported with regular updates.

1.4.1. Installing on Linux

Installing MQEdit on Linux is very easy. You just need to follow the following steps:

- **Installing WINE**
  MQEdit is a 32-bit Windows program and as such requires the 32-bit WINE package to be installed. There is a good chance that your Linux machine already has WINE installed but if it doesn't a simple command such as the following will likely do the job.

  ```
  sudo yum install wine.i686
  ```

  Of course, as is the nature with Linux, different distributions are often slightly different. Refer to your system documentation and the WINE documentation to discover the best way to install WINE.
- **Installing MQEdit**
  Download or copy the MQEdit zip file to your chosen Linux directory. Now unzip the file making sure to maintain the zip file directory structure.
- **Installing the IBM MQ interface libraries**
  MQEdit is a Windows program and yet we need it to call the Linux MQ libraries (libmqm_r.so or libmqic_r.so). This is done by installing MQ wrapper libraries (mqm.dll.so and mqic32.dll.so). These libraries can be downloaded from www.mqgem.com. They are contained in a zipped tar file **mq_wine.tgz**. Untar the file using a command such as

  ```
  tar -xvzf mq_wine.tgz
  ```

  You need to put these libraries where WINE will find them. Your two choices are:
  - Put them in /usr/lib/wine
  - Put them where ever you like but ensure that the path is added to WINEDLLPATH environment variable.

1.4.2. Running under Linux

Assuming that you have followed all the advice in 'Installing on Linux' then invoking MQEdit should be a simple matter of entering the command:

```
wine mqedit [ MQEdit parameters ]
```

Once you are comfortable that MQEdit starts and runs correctly you may want to set up an icon on your desktop to issue this command and possible run it in the background.

File Names

When running on Linux you can either use the normal Linux way of referring to OS files or you can use the Windows file format as described in the WINE File Management documentation.

1.4.3. Support

Each version of Windows behaves slightly differently and WINE is just the same. Although the folks at WINE have done a fabulous job of implementing the Windows interface there are bound to be subtle differences. In some cases these differences can be allowed for in the MQGem products and in some cases they will surface as a different behaviour. However, in general WINE can be treated by MQGem as just another Windows release. If you see a behaviour that you believe is not correct then by all means send us an email and we'll see what we can do to fix the problem. If necessary we can also contact the WINE development team and ask them to look in to the issue. Look on our web site for the current list of known problems.

---

[2] It also allows Mac OSX and BSD.

Chapter 2. Licensing

To access all the features of MQEdit you will require a licence file. The licence file also entitles the user to email support. If you would like to try out MQEdit for free then a 1-month trial licence can be obtained by sending a note to support@mqgem.com.

Each licence is for a certain period of time, usually one year rather than for a particular version of MQEdit. There are number of advantages of this scheme:

- **Customers of MQEdit get more dedicated support**
  Resources can be targeted towards customers who have made a financial contribution to the development of MQEdit.
- **Purchasing decision is simpler**
  The MQEdit licence covers a period of time not a release. It is therefore not necessary to concern oneself about whether a bigger, better version is about to come out soon since whatever licence you buy now will also work for that version.
- **Features are available sooner**
  Using this model it is not necessary for us to collect a large group of features together to 'justify' a new release of MQEdit. Instead a new release can be made available whenever a new feature is added which is regarded as sufficiently useful since all current users will be able to migrate to the new version at no cost to themselves.

There are five types of licences which allow different levels of flexibility about who can run the program. Essentially this is controlled by the presence, or not, of userid, machine or location fields.

Type	Fields Set	Description
Emerald	userid, machine	MQEdit is only supported on one machine using one userid.
Ruby	machine	MQEdit is supported by any number of users using it on the same machine.
Sapphire	userid	MQEdit is supported on any number of machines using the same userid.
Diamond	location	MQEdit is supported by any number of users at the same site on any set of machines. The location field gives the location, for example “London, England” of where the licence is based.
Enterprise	None	MQEdit is supported across your whole enterprise. This means any number of users at any number of locations. An Enterprise licence can be bought by purchasing just 3 Diamond licences.

2.1. Licence File Location

If a licence file is bought you will be sent an **mqgem.lic** file. All you need to do is place this licence file in the same directory as the mqedit.exe program. It is not necessary to end the program when installing a new licence file.

Alternatively you can set environment variable MQGEML to point to the directory where the licence file can be found.

2.2. Multiple licences

If you have multiple licences then they can be concatenated into a single **mqgem.lic** file. This can be done using simple OS commands such as **copy** or by using your favourite editor. Ensure that all lines are copied in their entirety, including the carriage return character at the end. Blank lines can be added to the file in required.

2.3. Licence Renewal

Licence renewal is never automatic. MQEdit will start issuing warning messages as you get close to the licence expiry date. These reminders will be in the form of pop-up dialogs.

By default MQEdit will start to remind the user about an expiring licence some 30 days before it expires. However, you can adjust this interval, if you like, using the environment variable, MQGEMLR. Set this environment variable to the number of days notice you want prior to expiry. For example **MQGEMLR=60** will request that MQEdit starts to notify of licence expiry 60 days before it expires. Setting the value to 0, **MQGEMLR=0**, will stop MQEdit issuing any licence expiry warnings. If you choose to do this, make sure you have some other way to remember when your licence will expire, to ensure that you have continuous licence coverage.

A new licence can be purchased at any time and a new licence file will be sent which extends the current licence by whatever period has been purchased. There is therefore no concern about losing time by renewing early.

2.4. Changing your licence file

The licence file is a simple text file. Generally speaking if you change the contents of the licence file you will invalidate it and it will cease to work. However, there are some minor changes you can make if you wish. Naturally it is always recommended that you keep a copy of the original unchanged file.

- You can change the case of any of the values.
- You can add or remove white space such as blanks
- You can add or change any lines which start with '*' since these are comment lines.

Remember that more than one licence can be contained within the single **mqgem.lic** file if required.

Chapter 3. Getting Started

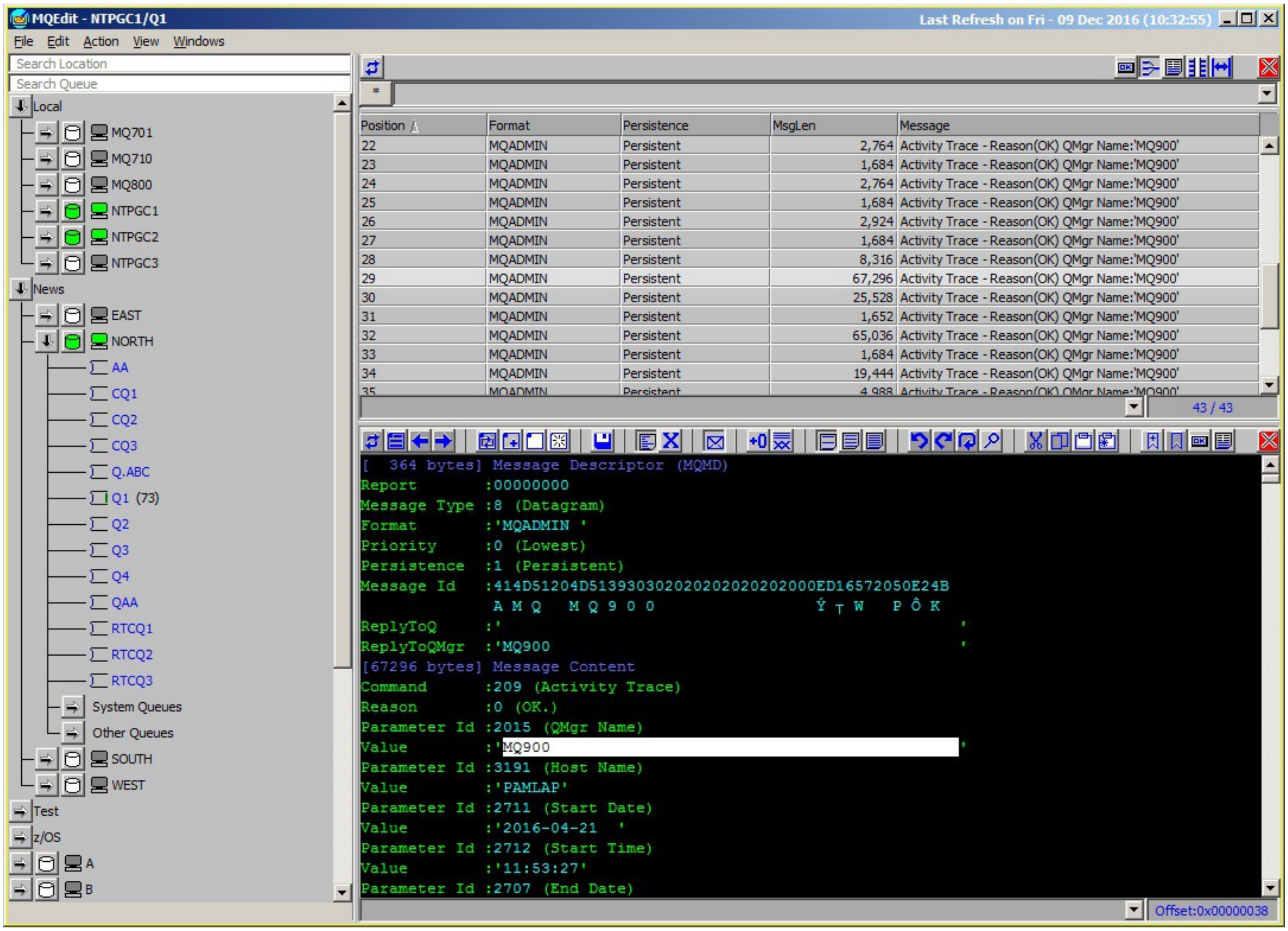
3.1. Running the Editor

All that is needed to run the editor is to run the mqedit.exe program. However, you do need to ensure that the correct MQ environment is set-up before the program is run. If you have installed MQ in the non-default location or you have multiple MQ installations then you will need to ensure the right installation is in effect before you run the mqedit.exe program.

There are essentially two ways you can start MQEdit depending on your MQ installation

1. **If you have a single MQ installation in the default location**
   You should be able to run MQEdit by issuing the command '**mqedit**' or double clicking on the **mqedit.exe** in the Explorer window.
2. **If you have multiple MQ installs or MQ is not at the default location**
   In this case you have to set-up the right MQ environment before calling MQEdit. The simplest way of doing this is to call the MQ command 'setmqenv'. To this end we have provided you with a simple command file called **mqeditstart.cmd**. Edit this command file to change the installation directory to the location of MQ installation you want to MQEdit to use. You should now be able to start MQEdit by issuing the command '**mqeditstart**' or double clicking on the '**mqeditstart.cmd**' in the Explorer window.

Once you have verified that your chosen method correctly starts MQEdit then you may find it convenient to add a desktop or Quick launch icon. This is achieved very simply by dragging either the mqedit.exe or mqeditstart.cmd to the location where you wish there to be a start icon. The program does not require any parameters but you can pass certain parameters if you wish. See *"MQEdit Parameters"* on page 138 for more information.



Position	Format	Persistence	MsgLen	Message
22	MQADMIN	Persistent	2,764	Activity Trace - Reason(OK) QMgr Name:'MQ900'
23	MQADMIN	Persistent	1,684	Activity Trace - Reason(OK) QMgr Name:'MQ900'
24	MQADMIN	Persistent	2,764	Activity Trace - Reason(OK) QMgr Name:'MQ900'
25	MQADMIN	Persistent	1,684	Activity Trace - Reason(OK) QMgr Name:'MQ900'
26	MQADMIN	Persistent	2,924	Activity Trace - Reason(OK) QMgr Name:'MQ900'
27	MQADMIN	Persistent	1,684	Activity Trace - Reason(OK) QMgr Name:'MQ900'
28	MQADMIN	Persistent	8,316	Activity Trace - Reason(OK) QMgr Name:'MQ900'
29	MQADMIN	Persistent	67,296	Activity Trace - Reason(OK) QMgr Name:'MQ900'
30	MQADMIN	Persistent	25,528	Activity Trace - Reason(OK) QMgr Name:'MQ900'
31	MQADMIN	Persistent	1,652	Activity Trace - Reason(OK) QMgr Name:'MQ900'
32	MQADMIN	Persistent	65,036	Activity Trace - Reason(OK) QMgr Name:'MQ900'
33	MQADMIN	Persistent	1,684	Activity Trace - Reason(OK) QMgr Name:'MQ900'
34	MQADMIN	Persistent	19,444	Activity Trace - Reason(OK) QMgr Name:'MQ900'
35	MQADMIN	Persistent	4,988	Activity Trace - Reason(OK) QMgr Name:'MQ900'

```
[  364 bytes] Message Descriptor (MQMD)
Report       :00000000
Message Type :8 (Datagram)
Format       :'MQADMIN '
Priority     :0 (Lowest)
Persistence  :1 (Persistent)
Message Id   :414D51204D513930302020202020202000ED16572050E24B
              A M Q   M Q 9 0 0                 Ý T W   P Ô K
ReplyToQ     :'                                                '
ReplyToQMgr  :'MQ900                                           '
[67296 bytes] Message Content
Command      :209 (Activity Trace)
Reason       :0 (OK.)
Parameter Id :2015 (QMgr Name)
Value        :'MQ900                                           '
Parameter Id :3191 (Host Name)
Value        :'PAMLAP'
Parameter Id :2711 (Start Date)
Value        :'2016-04-21  '
Parameter Id :2712 (Start Time)
Value        :'11:53:27'
Parameter Id :2707 (End Date)
```

Figure 1: Main Window

The main window is split into three sections.

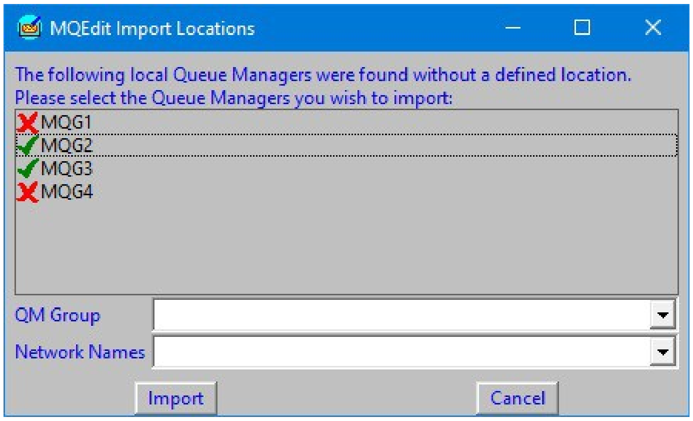
1. The left-hand pane shows the configured Queue Managers and their Queues
2. The top right-hand pane shows the messages on the selected queue
3. The bottom right-hand pane shows the message being edited.

It is often useful to be able to see as much of the message that is being edited as possible. So, by double-clicking on the edit pane it will expand to fill the entire window area. Double-clicking on it again will revert back to the three pane display.

Of course, when you first start the editor there will be no configured queue managers so let's first consider how those get created. We suggest that you start off with a test queue manager where you can play with the editor and get used to how it operates and it's features before you work on a live system.

3.1.1. Importing local Queue Managers

By default, each time MQEdit starts it will check whether all of the local Queue Managers on the Windows machine are defined. In the editor. If there are some that aren't defined a dialog will be shown which will give the user the opportunity to import definitions to each of the Queue Managers. For example, you could be shown a dialog like the one below:



The following local Queue Managers were found without a defined location.
Please select the Queue Managers you wish to import:

☒ MQG1
✓ MQG2
✓ MQG3
☒ MQG4

In this example we can see that four Queue Managers have been detected which are not defined in the MQEdit configuration. Note that importing a definition is doing nothing more than making a Location definition in MQEdit to access that Queue Manager. No MQ objects are imported, the Queue Manager itself is not accessed at this time.

If presented with this list the user can now decide which, if any, of these Queue Managers should be imported. Each entry is shown with a check (tick) mark or a cross. Only the entries with check (tick) mark will be imported. Clicking on the entry will change between the two.

The other things a user can specify (by typing in a name, or by selecting it from the drop down) is the name of the QM Group, and the set of Network Names, which should be used for the imported Queue Managers. The QM Group is merely a grouping concept in the Queue Manager pane of MQEdit. For example, you may wish to have all your local Queue Managers in a group called 'Local' as in this example. This can make finding particular Queue Managers much easier, particularly if you have a lot of them. Network Names are useful when searching for queue managers.

When the selection is correct the user should press the 'Import' button to have the location definitions automatically created. Conversely by pressing 'Cancel' none of the Queue Managers will be imported.

The imported location definitions will naturally contain default values for the various options you can specify for a location. If you wish to change these values then you can select 'Open location...' against each of the locations to make the modifications.

By default each time MQEdit starts the local Queue Manager check is made. If required this check can be suppressed in the 'General' tab of the Preferences Dialog. Just ensure that the 'Import local Queue Managers on start-up' is unchecked.

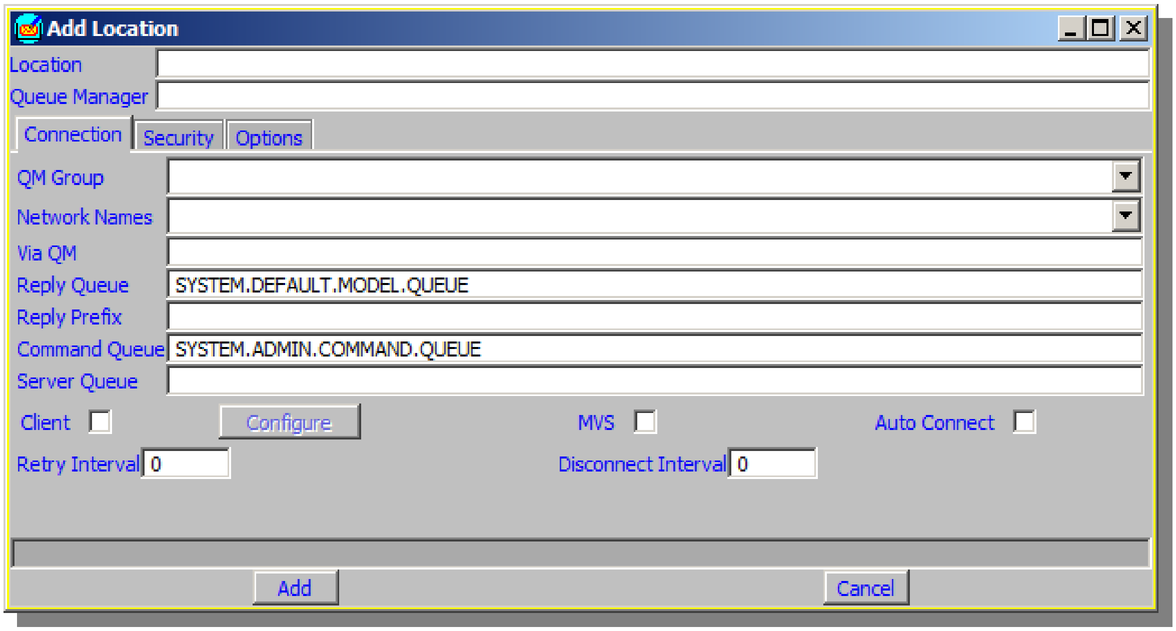
You can re-drive this import check at any time from the menu File → Import Local Queue Managers …

You can also import location definition from...

- a Client Channel Definitions Table (CCDT). Please see “From CCDT... ” on page 98 for more information.
- An MQ Explorer export file. Please see “From MQ Explorer File...” on page 99 for more information.
- another MQEdit Configuration file. Please see “From MQEdit Configuration...” on page 98 for more information.

3.2. Adding a Queue Manager Entry

To add an entry for a new Queue Manager in the Editor use *'Add Location'* in the File menu. This will bring up a location dialog.



Add Location

Location

Queue Manager

Connection | Security | Options

QM Group

Network Names

Via QM

Reply Queue: SYSTEM.DEFAULT.MODEL.QUEUE

Reply Prefix

Command Queue: SYSTEM.ADMIN.COMMAND.QUEUE

Server Queue

Client ☐

MVS ☐

Auto Connect ☐

Retry Interval: 0

Disconnect Interval: 0

Figure 2: Add Location Dialog

You can add a local Queue Manager by simply entering its name in the Queue Manager field and entering a text description of the location in the location field. The Queue Manager can either be local or connected to via a client. By default the application receives replies from the Queue Manager via the standard model queue SYSTEM.DEFAULT.MODEL.QUEUE but it may be preferable to define an explicit queue for MQEdit to use. See *"Reply Queue Details*" on page 128 for further details.

The following examples will show the additional configuration needed to use either of these methods.

3.2.1. Connecting via a Client

To connect to the Queue Manager via a client, select the Client check box and click on the Configure button to bring up a dialog to define the client connection channel. In this dialog fill in at least the Channel Name and the Connection Name.

For details of all the fields in the location dialog see "Location Dialog" on page 116.

Once the location has been added the Queue Manager location will appear in the left-hand pane. We now want to verify that we can connect successfully. First ensure that the Queue Manager is running and any required channels and listeners are available. Then click on the Queue Manager name. This should show a dialog displaying the Queue Manager definition and the Queue Manager icon should turn green indicating you are connected. If the connection fails then the icon will show a red cross. Opening the location dialog will show a message explaining the reason for the failure. If it is an MQ reason code, for example, '2059 Qmgr not available' then look at the MQ error logs to determine the cause of the failure. Once you have a valid connection then we can continue.

3.2.2. Creating two locations for the same Queue Manager

It should only be necessary to do this in very rare circumstance however it is possible. The reasons you might wish to do this are:

1. **In your MQ estate you have multiple Queue Managers with the same name**
   Of course every piece of MQ literature will strongly advise against having two different Queue Managers with the same name. Sometimes the reasons are understandable – such as we wish to have a test system that is identical to our production system. Sometimes the reasons are even humorous – the manual said we should issue CTRMQM QM1 – so we did! Either way, if you are in this unfortunate position rest assured, you are not alone.
2. **You wish to have multiple connections to the same physical Queue Manager**
   This can be useful if, for example, you wish to copy or move large numbers of messages via MQEdit and still be able to take other actions while that is going on.

Of course the simplest solution is just to run multiple instances of MQEdit. This is the recommended approach if you are trying to deal with the Test systems and Production systems having the same names problem. If this is the case you don't want anything connecting to both a production system **and** test systems; the possibility of cross-talk is just too high.

However, let's assume that you need to create multiple locations to the same Queue Manager – how do you do it? Well, the basic rule is that the combination of the Queue Manager name and Reply Queue must be unique. This means we have two options – we either change the Queue Manager name or we change the Reply Queue name.

- **Changing the reply queue**
  This is pretty straight-forward. The simplest approach is just to define a second model queue which can be used by these 'other' connections.
- **Changing the Queue Manager name**
  'How?' I hear you ask "can you possibly change the Queue Manager name and still connect to the right one?"'. Well, you can use a feature of the MQ Client that any Queue Manager name starting with an asterisk(*) will be allowed to connect to a Queue Manager of any name. In other words if the Queue Manager starts with '*' it will allow the Queue Manager name to be different than the actual Queue Manager name. So, suppose you have a Queue Manager called 'QM1' you could define a location and specify the Queue Manager as '*QM1_SECONDARY'. Exactly what you call it does not matter, as long as it starts with an asterisk (*). Note though that this method only works if you are connecting via a client. You could still use it for a local Queue Manager it's just that you have to connect via the client library.

3.3. Accessing the list of Queues

The next step is to display the list of queues. This is simply a question of clicking on the arrow symbol to the left of the Queue Manager. This will now show the list of local queues. Below these there are two other twisties, one containing any defined system queues and another containing all the other defined queues. The reason for this separation should be clear. A message editor is mainly concerned with messages. Since messages can only actually reside on local queues it is those queues we are mainly concerned with. However, you may want to interact with other queues types, such as alias queues, as we shall see later so they have been included.

So, now we have a list of queues. The next thing to do is to get a list of the messages on a queue. Choose the queue you are interested on and double-click on it. You will see that the top right-hand pane now shows a list of messages. By default only the first 100 messages are shown but you can change this in the preference dialogs if you wish. Alternatively you can change the 'Display Range' field and then refresh the list.

Depending on the version of Windows and your personalize settings the editor may show the last refreshed time in the title bar of the window. This time will be updated each time you refresh the list or, indeed, when it is auto-refreshed. You can change the format of this display to some extent. Please see Chapter19.11.1 'Large Format, Medium Format, Small Format' on page 115 to see how you can do this.

3.4. Accessing an individual message

As you may have guessed, accessing an individual message is merely a question of double-clicking on the message you wish to edit or display. This will display the message in the bottom right-hand pane. You may find it more convenient to double-click on this pane to expand it to full size while you edit the message.

The editor can display the message in two formats. By default it will try to display the message in a formatted display. The editor knows various MQ formats and headers, such as PCF messages, MQRFH2, Transmission Queue and Dead Letter headers. If the format is unknown then the message will be displayed in hex. Generally speaking editing a message while it is formatted is far simpler[3].

The message editor is a 'live parsing' editor. This means that as you edit the message the display is reformatted in real time. Not only that but MQEdit will automatically adjust the required length fields which makes editing a breeze.

Consider a PCF message. A PCF message is a message that can contain many sub-structures that contain various data types. For example, there are structures that contain integers, strings, arrays, byte strings etc. It is a very flexible message format and fairly efficient to parse and is therefore widely used by both the IBM MQ product itself (in event and activity messages for example) and by MQ users. However, PCF messages are a non-trivial format. Creating a PCF message in a program is fairly simple but editing a PCF message has, until now, been very hard. The reason is that these sub-structures have length fields which must be exactly right for the message to be parsable. Getting all these lengths right can be a laborious and error prone task. MQEdit removes all this complexity. All you need do is to add or delete text to your strings and MQEdit will update the lengths of the various fields as necessary.

---

[3] Over time we hope to support more and more message formats. If you have a message format which you think ought to be formatted but is unknown to MQEdit then let us know and perhaps we can add it.

3.4.1.1. Use of Message Token

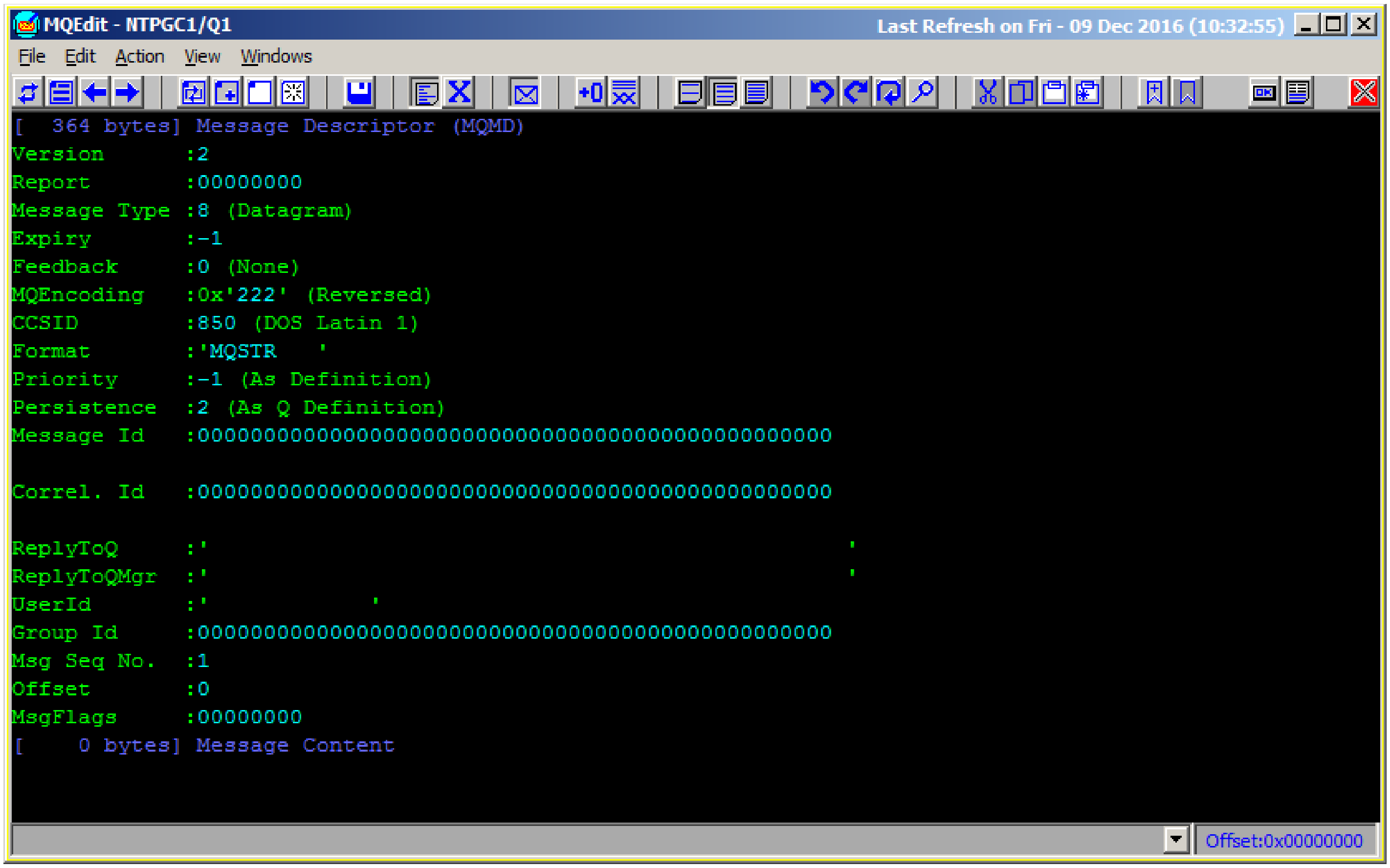
The editor uses the Message Token to access a particular message when going from the message list to a message in the edit window. This has the advantage that Message Token is guaranteed to be unique, where as Message Id is not.

However, the Message Token will change if the Queue Manager is ended and started again. So, in the unlikely event that you are using the editor while the Queue Manager is cycled then you may need to refresh your message list to ensure that the message tokens are valid again.

3.5. Creating our first message

To see how easy this is let's try a worked example. Let's create a PCF message containing a string of text from scratch. The first thing we want to do is tell MQEdit that we want an 'empty' message. Many of the commands can be accessed in three ways. Firstly you can click the tool in tool bar. Secondly you can select the command from the context menu displayed when you right-click in the window. Lastly, if you have dialog buttons displayed, you can press the required button. Different users have their own preferences so use the method you are most comfortable with. In general we will refer to the toolbar but if you prefer by all means use an alternative method.

So, press the 'Empty' tool from the toolbar. Hovering the mouse over the toolbar buttons will tell you what each button does. Ensure that the Message Descriptor tool is depressed and select the 'medium' level of detail. MQEdit displays something like this:



```
[  364 bytes] Message Descriptor (MQMD)
Version      :2
Report       :00000000
Message Type :8  (Datagram)
Expiry       :-1
Feedback     :0  (None)
MQEncoding   :0x'222'  (Reversed)
CCSID        :850  (DOS Latin 1)
Format       :'MQSTR   '
Priority     :-1  (As Definition)
Persistence  :2  (As Q Definition)
Message Id   :000000000000000000000000000000000000000000000000

Correl. Id   :000000000000000000000000000000000000000000000000

ReplyToQ     :'                                                '
ReplyToQMgr  :'                                                '
UserId       :'            '
Group Id     :000000000000000000000000000000000000000000000000
Msg Seq No.  :1
Offset       :0
MsgFlags     :00000000
[    0 bytes] Message Content
```

You can see that we have essentially a 'blank' Message Descriptor. However, there are two things to notice about the values.

1. **The default format is MQSTR**
   By default MQEdit will assume you just want to create a simple string message.

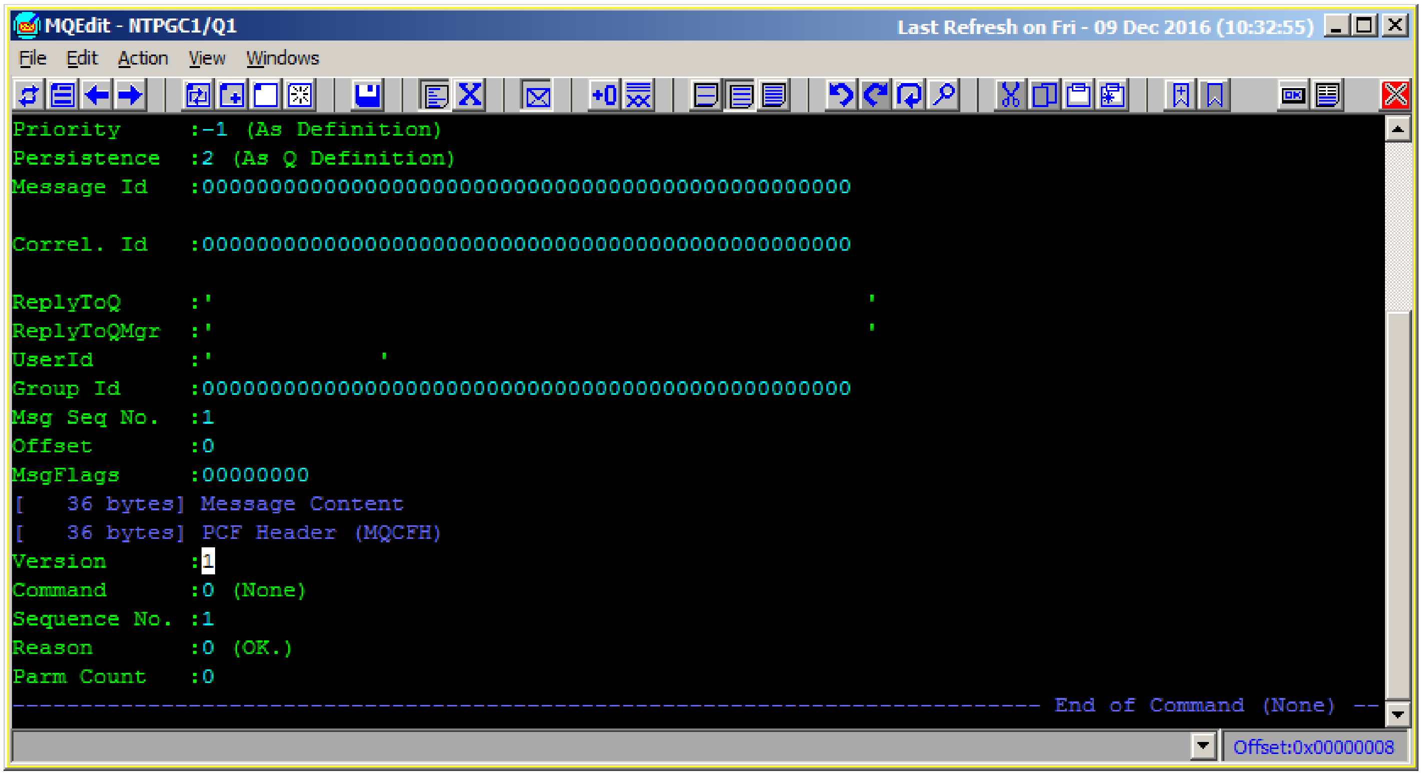
2. **The default codepage and encoding values**
   MQEdit maintains a default encoding and codepage for each Queue Manager. You may wish to change these. This can be done by changing the Default Codepageand Default Encoding fields in the Options Tab tab in the Location Dialog.

Feel free to change any of the Message Descriptor fields. However, remember that only certain Message Descriptor fields can be set unless you are using 'Set All Context'. By default MQEdit will use Default context so all the application and origin context will be generated by IBM MQ itself.

Of course at the moment we are more interested in the message itself. Clearly we could just start typing in the message field and create ourselves a string message. However, we said we wanted a PCF message. So how do we create one of those?

Well we simply need to put the cursor at the start of the message and right-click on the window. This brings up the context menu, one of which is 'Add Message Part'. Use this menu and navigate to 'PCF Header (MQCFH)' and click it. You can see that there are a fair number of things we could have added and you can play with these later, for now we shall see what happened.

Our screen now looks something like this:

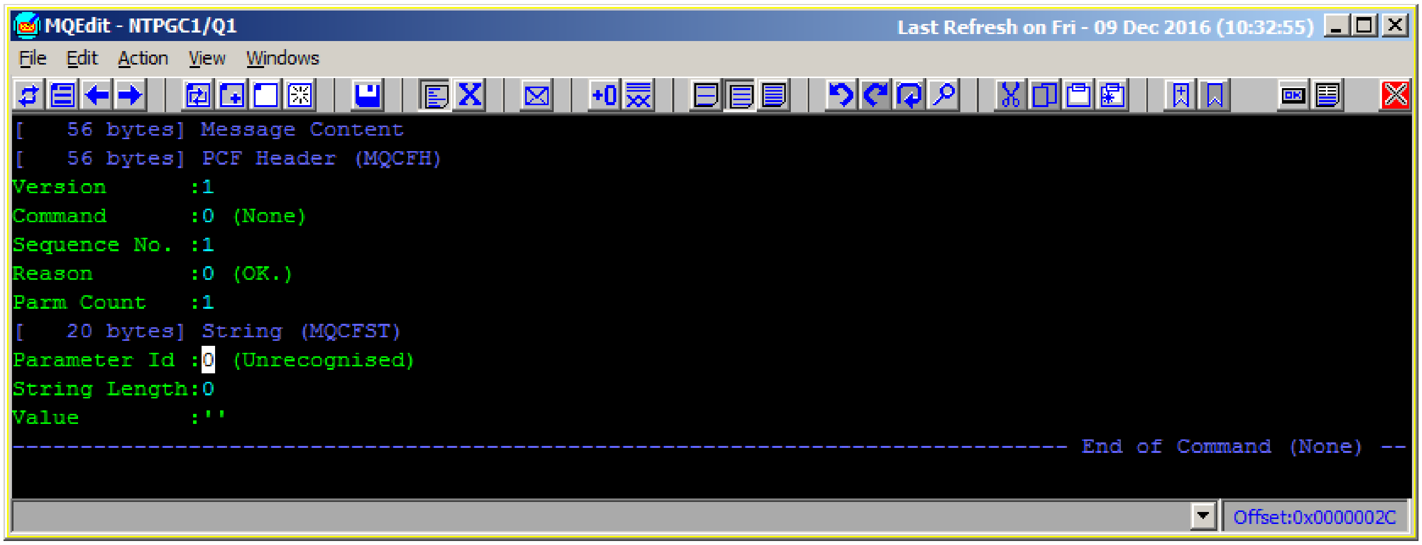


```
Priority      :-1  (As Definition)
Persistence   :2  (As Q Definition)
Message Id    :000000000000000000000000000000000000000000000000

Correl. Id    :000000000000000000000000000000000000000000000000

ReplyToQ      :'                                               '
ReplyToQMgr   :'                                               '
UserId        :'            '
Group Id      :000000000000000000000000000000000000000000000000
Msg Seq No.   :1
Offset        :0
MsgFlags      :00000000
[   36 bytes] Message Content
[   36 bytes] PCF Header (MQCFH)
Version       :1
Command       :0  (None)
Sequence No.  :1
Reason        :0  (OK.)
Parm Count    :0
----------------------------------------------------------------------------- End of Command (None) --
```

Selecting that menu item has clearly changed things. Most notably we now have the standard MQCFH structure at the start of the message. We can also see that the format of the message in the Message Descriptor has changed to MQPCF.

We can see that the Undo tool is now enabled. Of course we are quite happy with the change we have made but just to try it out let's press the Undo tool. You can see that the message and message descriptor goes back to how it was. Perhaps not a surprise but it's nice to know it's there. If ever you are editing a message and things 'go wrong' then you can get back on track by pressing Undo. Likewise you can press Redo to go forward again, let's do that now.

So, now we want to add a string to the PCF message. It won't surprise you to learn that this is very similar action to what we did before. Put the cursor somewhere in the MQCFH structure and right-click on the window. Now, navigate the 'Add Message Part' menu until we are adding a String MQCFST. The screen now looks something like this.

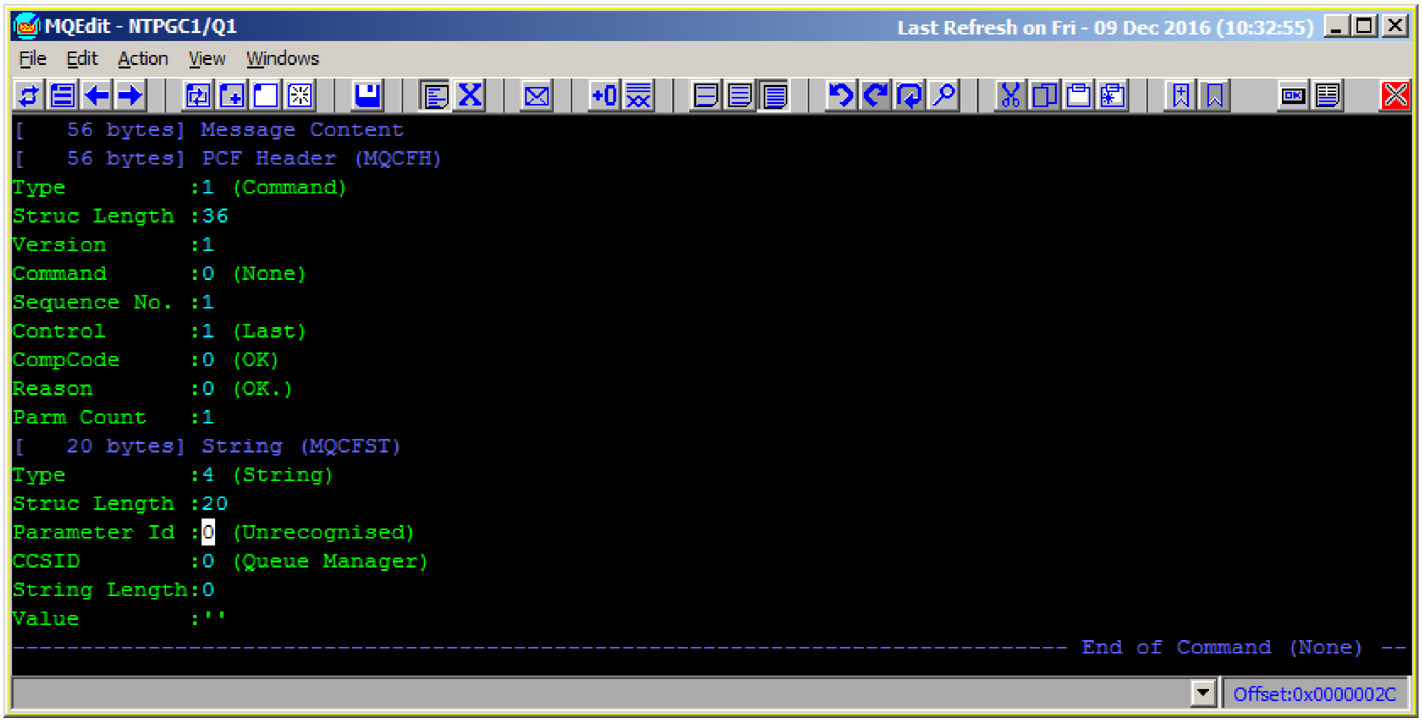


```
[   56 bytes] Message Content
[   56 bytes] PCF Header (MQCFH)
Version       :1
Command       :0  (None)
Sequence No.  :1
Reason        :0  (OK.)
Parm Count    :1
[   20 bytes] String (MQCFST)
Parameter Id  :0  (Unrecognised)
String Length:0
Value         :''
------------------------------------------------------------------------------- End of Command (None) --
```

Notice that we've hidden the Message Descriptor (another tool bar button) for now just to reduce the size of the window.

One of the things to notice here is that the 'Parm Count' field has changed to 1. The 'Parm Count' field is the value in the MQCFH structure which says how many parameter structures follow the MQCFH. MQEdit has updated this automatically for you because you added a new structure. You don't need to worry about it.

Of course those that know the PCF message format will notice that we're not seeing all the structure fields. Let's fix that by showing full message detail (another toolbar button).

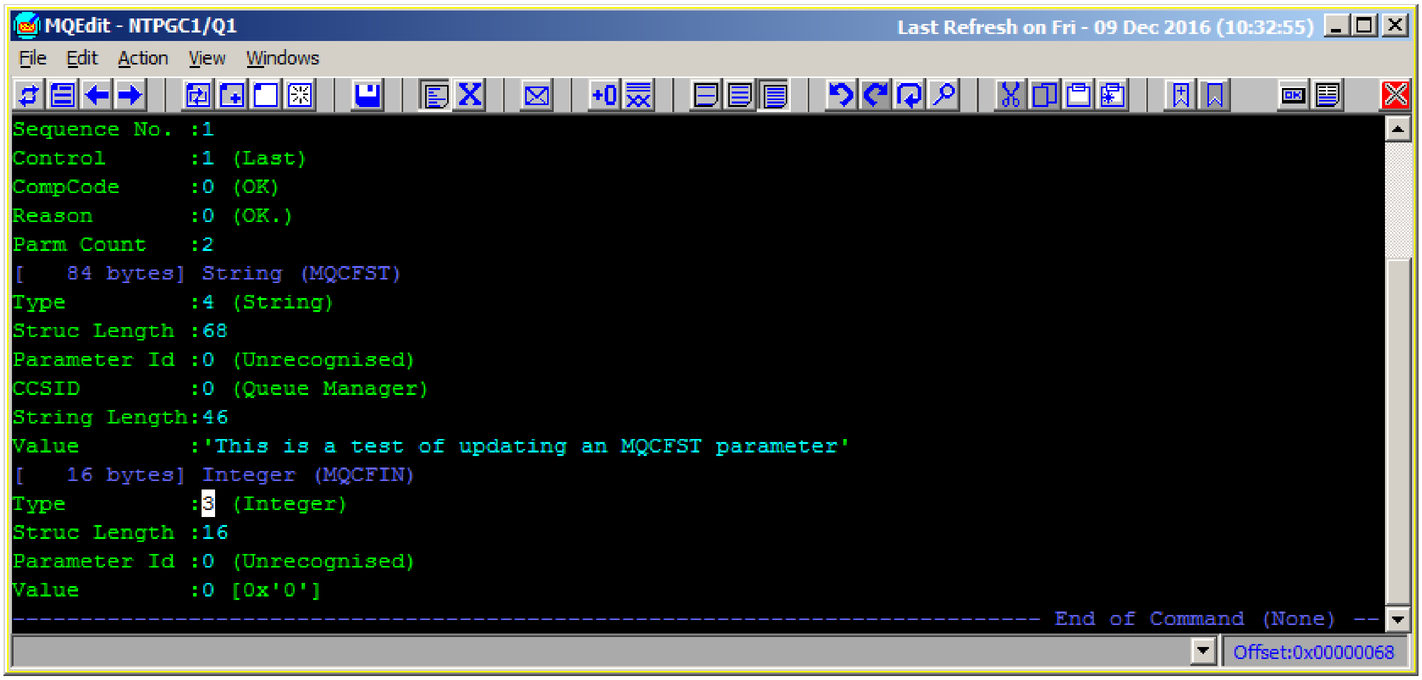


```
[   56 bytes] Message Content
[   56 bytes] PCF Header (MQCFH)
Type         :1  (Command)
Struc Length :36
Version      :1
Command      :0  (None)
Sequence No. :1
Control      :1  (Last)
CompCode     :0  (OK)
Reason       :0  (OK.)
Parm Count   :1
[   20 bytes] String (MQCFST)
Type         :4  (String)
Struc Length :20
Parameter Id :0  (Unrecognised)
CCSID        :0  (Queue Manager)
String Length:0
Value        :''
------------------------------------------------------------------------------ End of Command (None) --
```

Now we can see all the fields in the structure. For example, we now see the lengths of the structures. Now let's enter some text in the string field and see what happened to the lengths.

You can see that as text is added or removed from the string that both the string length changes as well as the structure length. Have a play with adding and removing text. For example, you could copy and paste text from another document. Also try out the undo and redo buttons.

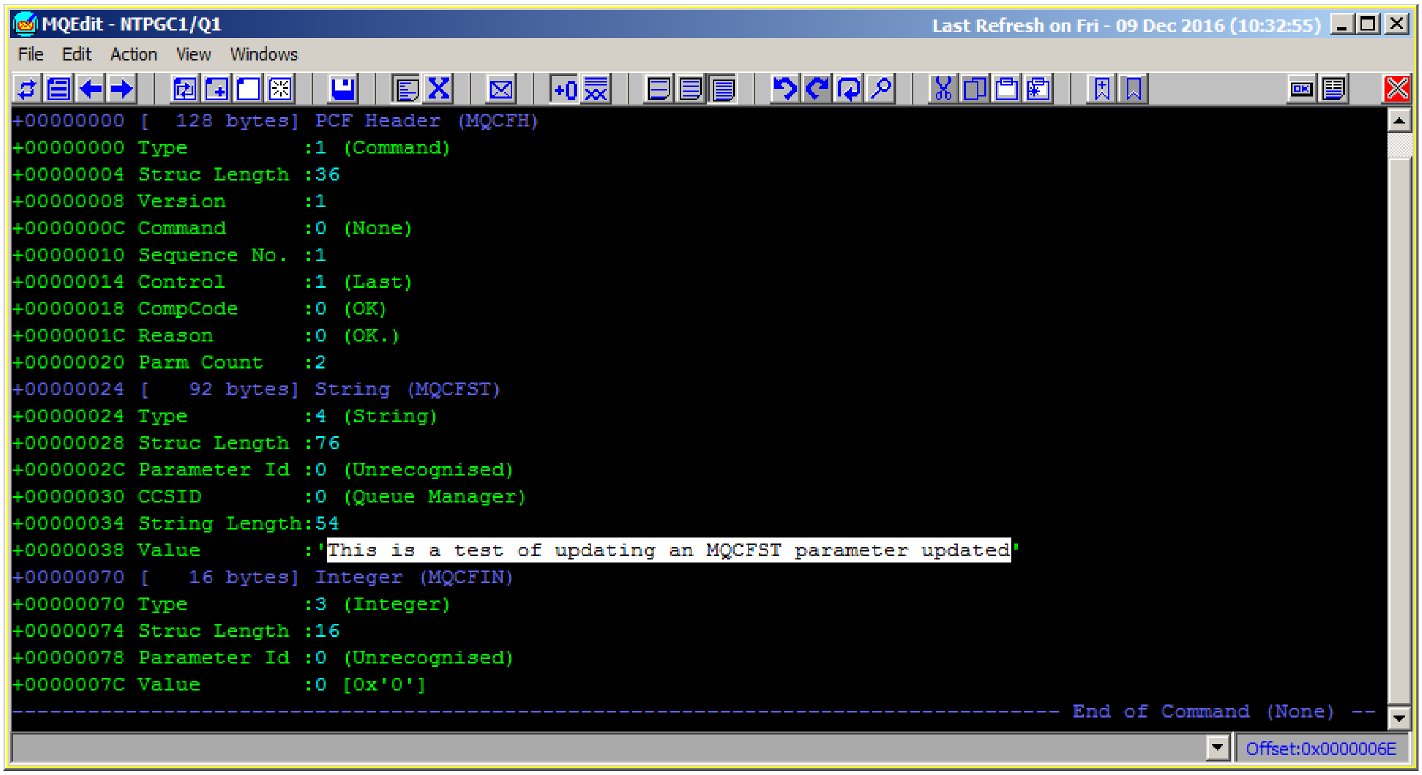
You can see that whatever text you choose to put in the string, the length fields are maintained automatically. This means that the PCF message structure stays intact. However, suppose there were other parameters fields after the string structure ? What would happen then ? Well, let's try it. Why not add an integer after the string. So now we have something a little like this.



```
Sequence No. :1
Control      :1  (Last)
CompCode     :0  (OK)
Reason       :0  (OK.)
Parm Count   :2
[   84 bytes] String (MQCFST)
Type         :4  (String)
Struc Length :68
Parameter Id :0  (Unrecognised)
CCSID        :0  (Queue Manager)
String Length:46
Value        :'This is a test of updating an MQCFST parameter'
[   16 bytes] Integer (MQCFIN)
Type         :3  (Integer)
Struc Length :16
Parameter Id :0  (Unrecognised)
Value        :0  [0x'0']
---------------------------------------------------------------------------- End of Command (None) --
```

Again notice that the 'Parm Count' field is incremented for you automatically.

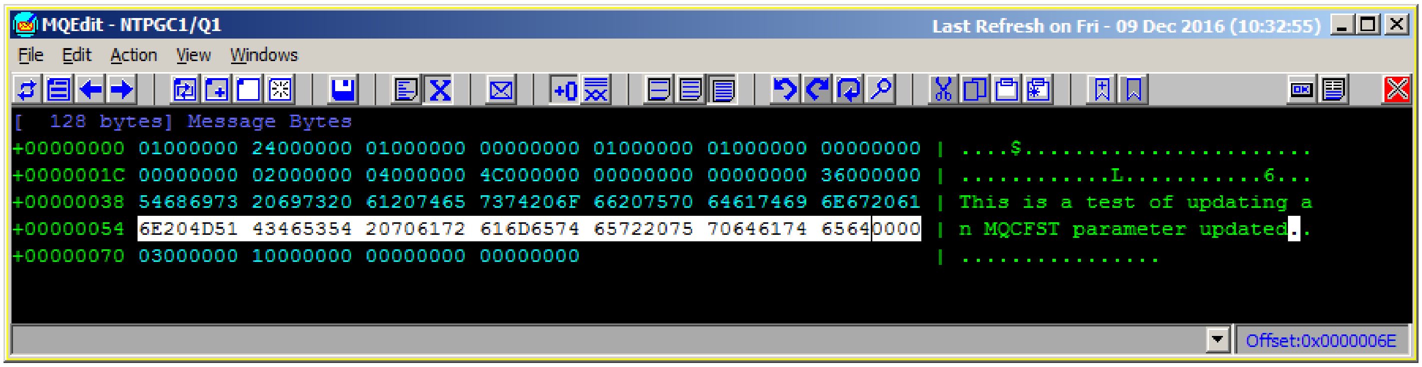
Now let's try modifying the MQCFST again. We can see that it's rather unexciting. The MQCFIN structure output doesn't change at all. It looks for all the world as though it is completely unaffected. Of course we know it is affected since the MQCFST structure length is changing. Each time the MQCFST length changes all the following structures much either be shuffled forward or shuffled backwards. However, MQEdit does this so quickly and easily that it's easy to forget that it is happening. One clue that is happening is to periodically put the cursor on the start of the MQCFIN structure and look at the bottom right of the screen. This small window tells you the offset in the message of the cursor position. So, we can verify that the offset of the MQCFIN structure changes as we add and remove data from the MQCFST structure. The simplest way to view the offsets of the structures and the fields within them is to press the offsets (+0) tool. This will change the display to something like this.



```
+00000000 [  128 bytes] PCF Header (MQCFH)
+00000000 Type         :1  (Command)
+00000004 Struc Length :36
+00000008 Version      :1
+0000000C Command      :0  (None)
+00000010 Sequence No. :1
+00000014 Control      :1  (Last)
+00000018 CompCode     :0  (OK)
+0000001C Reason       :0  (OK.)
+00000020 Parm Count   :2
+00000024 [   92 bytes] String (MQCFST)
+00000024 Type         :4  (String)
+00000028 Struc Length :76
+0000002C Parameter Id :0  (Unrecognised)
+00000030 CCSID        :0  (Queue Manager)
+00000034 String Length:54
+00000038 Value        :'This is a test of updating an MQCFST parameter updated'
+00000070 [   16 bytes] Integer (MQCFIN)
+00000070 Type         :3  (Integer)
+00000074 Struc Length :16
+00000078 Parameter Id :0  (Unrecognised)
+0000007C Value        :0  [0x'0']
--------------------------------------------------------------------------- End of Command (None) --
```

This can be a very handy way of keeping track of all the 'bits' of your message. Try changing the MQCFST string now. You will see that as you add and remove text the offset of the MQCFIN structure and its fields will change. Note that the position of the MQCFIN only changes every four bytes though since the length of an MQCFST structure is always rounded up to the next multiple of four bytes.

Of course the other way to view the full structure of the message is to look at the message in hex. Go ahead and click on the hex formatting button on the toolbar.



```
[  128 bytes] Message Bytes
+00000000 01000000 24000000 01000000 00000000 01000000 01000000 00000000 | ....$.......................
+0000001C 00000000 02000000 04000000 4C000000 00000000 00000000 36000000 | ............L...........6...
+00000038 54686973 20697320 61207465 7374206F 66207570 64617469 6E672061 | This is a test of updating a
+00000054 6E204D51 43465354 20706172 616D6574 65722075 70646174 65640000 | n MQCFST parameter updated..
+00000070 03000000 10000000 00000000 00000000                            | ................
```

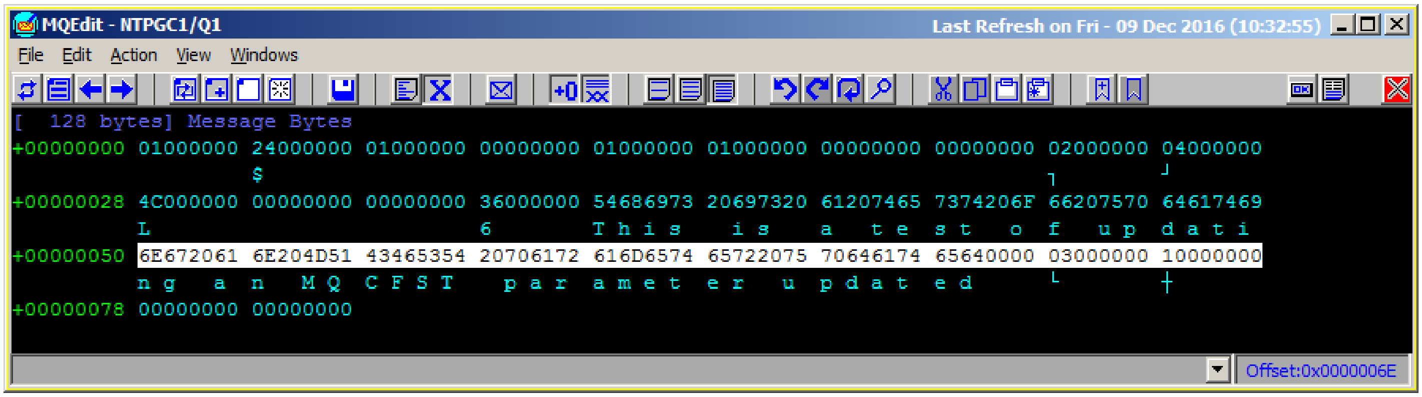
This display shows you all of the actual bytes that are contained in the message. On the right-hand side it also shows you what characters those bytes represent (if any). This can be a useful indicator of where in the message your cursor actually is. Note however that you cannot edit the characters on the right, they are just there for convenience or a memory jogger.

However, if we are feeling brave and our hexadecimal skills are up to scratch we can edit the hex bytes in this mode. However, bear in mind that in hex mode the data is treated just as a sequence of bytes. MQEdit will not try to apply any formatting on the data and therefore if you add extra characters in the string field it will not update any lengths for you. You can, of course, update them yourself. HEX mode is useful for making changes to a message that formatting doesn't help you with. For example, consider a message which is badly formatted. Let's say that it has one length field that is incorrectly byte swapped. You can't edit the message in formatted mode since the message is badly formed. So, you can go in to HEX mode, fix the offending length and then revert back to formatted mode. It is not intended that users will do extensive editing HEX mode but rather it is the fall back option. If nothing else will work for you then you can always go to HEX mode and make all changes by hand.

However, let's try a few edits in HEX mode just to see how it works. Put the cursor at the start of the string field. Notice that the highlighted square on the right-hand side shows you where you are. Now let's over-write with the characters 'MQGem'. Of course we need to do that in HEX. So, you actually need to type the characters 4D5147656D[4]. Remember though that you need to be in overwrite mode – so press the INSERT key until you have a block cursor. Of course you **can** insert the bytes if you wish, you just need to remember to delete the same number of bytes after the cursor position. Otherwise you will have changed the length of the string and you will need to update the string length and structure length fields. By the same token you could just paste in the bytes you want. For example, copy the hex string from this manual and paste it in to the message. However, pasting is always an insert operation so make sure you select 5 bytes (10 nibbles) before you paste the data in, otherwise again you are changing the length of the string and the format of the message will not be correct.

Even making these simple changes in HEX mode demonstrates how much is going on behind the scenes for you when you use MQEdit in formatted mode!

MQEdit also allows another way of editor a message in hex. Press undo until you have got back to the point just before you added the 'MQGem' text. Now press the 'Display Hex as chars' option. You should now see a screen that looks something like this:

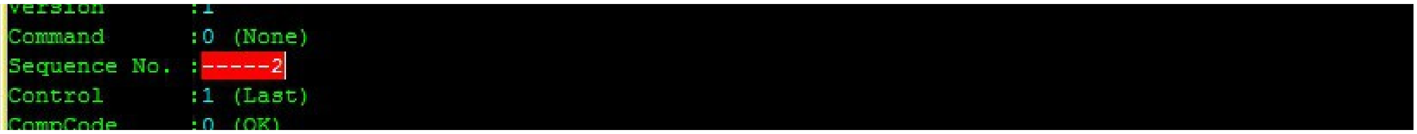


```
[  128 bytes] Message Bytes
+00000000 01000000 24000000 01000000 00000000 01000000 01000000 00000000 00000000 02000000 04000000
                   $                                                              ┐        ┘
+00000028 4C000000 00000000 00000000 36000000 54686973 20697320 61207465 7374206F 66207570 64617469
          L                          6        T h i s    i s    a   t e  s t   o  f   u p  d a t i
+00000050 6E672061 6E204D51 43465354 20706172 616D6574 65722075 70646174 65640000 03000000 10000000
          n g   a  n   M Q  C F S T    p a r  a m e t  e r   u  p d a t  e d      └        ┼
+00000078 00000000 00000000
```

You can see now that each hex line is actually displayed twice – once in hexadecimal and once in character format. So what is the difference between this and the previous display? Well the difference is that we can now edit the characters directly. We don't need to work out what the character bytes are, we can just put the cursor on a bit of character text and type away. Remember again that if you are not in overwrite mode you will be changing the actual format of the message since the structure lengths will be wrong. Try it, insert a few characters and then look at the message in formatted mode. Depending on what you type and where there is a good chance that the MQCFIN structure is now no longer displayed correctly. The bytes are still there, they are just in the wrong place. Remember that you can just press undo to get back to a consistent state.

OK, so that is hex mode, let's go back to formatted mode, it is so much easier!.

Now, we can see the PCF message again. Switch off the offset display too if you like since we don't normally care too much about the field offsets. Let's try editing some of the other fields and see what happens. Try putting the cursor on the 'Sequence No.' field of the MQCFH structure. We choose this field because it has no extra properties. It is about as boring as an integer field can be. You should see that as you edit the number MQEdit reflects the edits as you would expect, provided of course, that you enter a valid number. Suppose you enter an invalid number. An invalid number is one that doesn't match what would be displayed. For example, try entering -----2. You'll see that MQSC will display this:



```
Version       :1
Command       :0  (None)
Sequence No.  :-----2
Control       :1  (Last)
CompCode      :0  (OK)
```

---

[4] It is possible that you are testing this out as a client to an EBCDIC Queue Manager. If that is the case then you would need to type the EBCDIC characters for MQGem which would be D4D8C78594

You can see that MQEdit will highlight the value. This highlight is MQEdit's way of telling you that the value has not actually been changed in the data yet. The reason being that the value is not correct. There are other reasons that the number might be inappropriate. For example it could have leading zeros or it could be too large. MQEdit will always highlight the data in this way to tell you that the data hasn't actually made it in to the message. We can show that just by moving the cursor away from this field. You will see that the data will immediately revert back to the last valid number that was entered in to this field.

So, let's try editing one of the other fields and see what happens. Let's try the 'Struc Length' field.

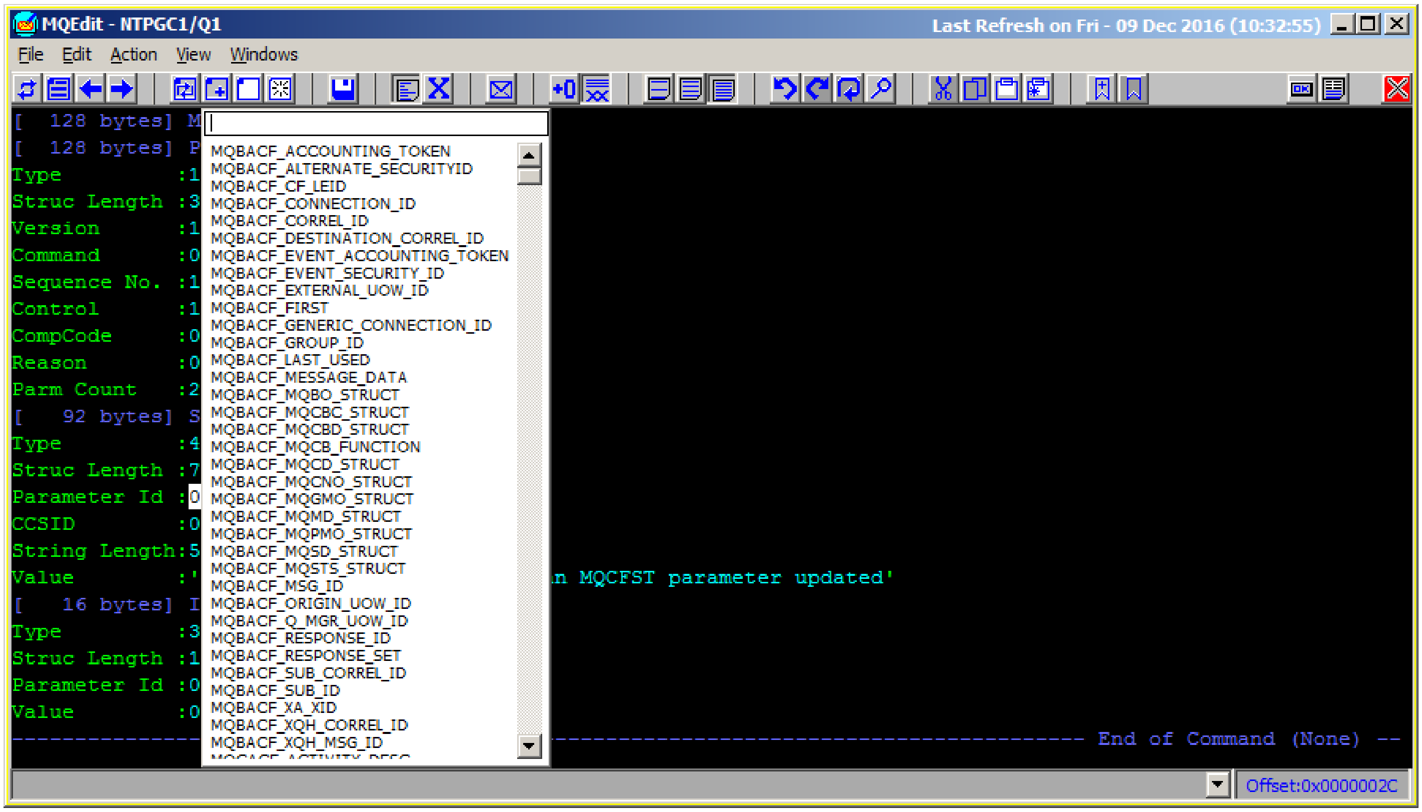
Did it work ? Can you change the number ? Probably not. MQEdit probably displayed a message at the bottom of the screen that says 'To edit this field you need to be in expert mode'. Why is this? Well, perhaps by now you can already tell. MQEdit it trying to be helpful here. There are certain fields which are generally better left to the editor to calculate for you automatically. If you change the value and get it wrong then the message formatting will fall to pieces. You can change the value if you really, really want to, by switching to expert mode (and we'll talk a little about that later) but for now let's just appreciate that some fields are better left to the editor if at all possible.

3.5.1. Selection Lists

So, now let's try editing the 'Parameter Id' field of the MQCFST structure. This field is a number which tells the receiver of this message what this parameter actually means. MQ defines lots of values for this field and there is a range that you can use in your application if none of the predefined values are suitable. Put your cursor on the field and start changing the integer. You'll notice that as you change the value the editor will display a small text string saying what the value means. For example, type in the value 2015 and you'll see it says 'QMgr Name'. This is because IBM MQ has defined that whenever a value of 2015 appears in a parameter structure then it represents a Queue Manager Name. Of course it can be a laborious process trying to find the right number for the field you want. What we would like is a way of having the list of known options presented to us. Luckily, we can do just that.

If you move the mouse slightly to the right of the field you will see a coloured box appear. Press it. Up pops a selection list. This will show you all the defined parameter ids and there a lot of them! The first thing to be aware of is that you need to keep your mouse in the box. As soon as the mouse moves away from the box it will disappear. This is a very quick way of showing and hiding the box. However, some people don't like the box auto hiding and would like to be more explicit about it. If that is you then try this. Put the cursor on the field you are interested in and press F4. The box is shown again, but this time, since you used a key stroke to show the selection list you have to press ESC to hide it again. Unless of course you make a selection which is simply a case of double clicking on the one you want.

Our screen now looks something like this:



```
[   128 bytes] M
[   128 bytes] P
Type          :1
Struc Length  :3
Version       :1
Command       :0
Sequence No.  :1
Control       :1
CompCode      :0
Reason        :0
Parm Count    :2
[    92 bytes] S
Type          :4
Struc Length  :7
Parameter Id  :0
CCSID         :0
String Length:5
Value         :'                                         n MQCFST parameter updated'
[    16 bytes] I
Type          :3
Struc Length  :1
Parameter Id  :0
Value         :0
-------------------------------------------------------------------------- End of Command (None) --
```

The first thing you'll notice is that there are a lot of values! The second thing you notice is that instead of a string description you are shown the IBM MQ constant. This is generally better than a string description because it matches the descriptions of the PCF messages and events in the IBM MQ manuals. However, if you really want a small text description then try bringing up the selection list while holding down the 'Shift' key.

Now, depending on what value was in your field when you brought up the selection list you may notice something else. And that is that the current value is highlighted in the list. However, you may need to scroll down the list to find it.

We can reduce the size of this list by entering a search string in the top entry field. For example, typing 'name' will show only those values that have the text name in them. We can now scroll down until we find MQCA_Q_MGR_NAME and double click the entry. This will put the value of 2015 in our field.

There is another type of selection list used for flag fields. Let's try that one out now. Change your display so that the Message Descriptor is shown and bring up the selection list for the 'Report' field.

You should see the selection box has multiple entries selected. Each of these represents a value in the bit flags. In other words, this type of field can have more than one entry selected. This makes sense, in the previous case we were looking for a single number, in this case we are matching against a set of values. As before we can go ahead and change the selection by click with the mouse or pressing the space bar. However, double-clicking no longer dismisses the dialog. This is because we are not looking for a single selection. We need another way of telling MQEdit that we are finished selecting values. All we need do is press the RETURN or ENTER key.

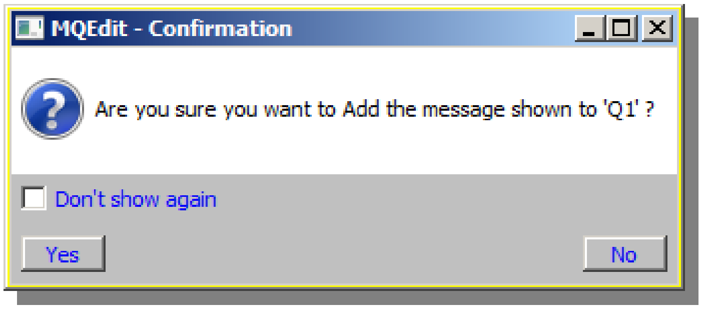
One other thing to mention about selection lists is that the entry field at the top isn't always shown. Sometimes there are so few choices it isn't really worth it. Try bringing up the selection list for the 'Message Type' field. You can see that the selection list is still really useful, we don't need to remember the message type values, but with only seven of them we don't really need a way of reducing the list! So, now you have the basics. You should be able to add and remove parts of the PCF message and edit all the fields presented. Once you are happy with the message then we can add it to the queue.

3.6. Adding our first message

Adding the message is simply a case of pressing the 'Add' tool. What ever the contents of the message and message descriptor are at this instant will be sent to the Queue Manager. However, before you press the 'Add' tool have a thought for what context you would like to use. A full description can be found in “Message Context“ on page 21. You can change the context option you want to use by right-clicking on the edit window and use the 'context – Add' menu.

For a more complete description and understanding of context fields you should refer to the MQ documentation. There are four context settings which are selectable via the context menu. Since we are adding a new message we will most likely want to use default context. This means the IBM MQ will fill in the context fields for us and it requires the least level of security. However, if you have updated the Message Descriptor context fields then you may will to use one of the other values.

So, now let's press the 'Add' tool. MQEdit will more than likely show you a confirmation dialog like this:



Are you sure you want to Add the message shown to 'Q1' ?

MQEdit will do this for any action that is potentially hazardous. Of course only you know whether it should really be considered hazardous or not so you have the option to not show the confirmation if you prefer. If you do click the “Don't show again” checkbox then rest assured that you can re-enable the confirmation by re-checking the option in the 'Confirmation' tab of the Preferences Dialog.

So, let's press 'Yes' and continue with the Add. Assuming that the Add operation works we will see a message at the bottom of the screen saying '1 message added' and the message list display is probably updated to show our new message on the queue[5].

Congratulation! You have just created a new message and added it to a queue.

3.7. Updating our first message

It won't surprise you to hear that updating a message is very similar to creating a new one. The first thing we need to do is to display one of the messages in the message list. So, double-click on the message we've just created (or choose another message if you prefer). The message is immediately shown in the bottom window. We can edit the message as we did before. When we are ready to update the message we can press the 'Update' tool.

Now, it is important you understand what an 'Update' actually does. Unfortunately IBM MQ does not support updating a message in place. The only way to change a message is to remove the original message and put a new message on the queue and this is exactly what MQEdit does. This has two consequences:

1. **The message is put back potentially at a different position in the queue**
   This doesn't usually matter, applications should not rely on a particular message order. However, if this message is part of a group of messages and your application is not well behaved then you may need to update the other messages in the group to maintain their relative order. This may be easier done on a separate queue. MQEdit makes it easy to move a group of

---

[5] Whether the message list is updated automatically can be set in the preference dialog.

messages to a different 'work' queue – you can even just drag&drop them! You can edit the set of messages until they are as you want them and then drag them back to the 'live' queue.

2. **The message will have a new message token**
   The editor uses the Message Token to uniquely identify a message. When you double click on a message list display it is the message token which is used as the key to retrieve the message. This allows MQEdit to be very fast in retrieving the message even if it is getting the 100,000th message from the queue. However, it does mean that when a message is updated you can not immediately update it again. This is because the Message Token that MQEdit has in it's hand is no longer valid. You need to re-select the message from the message list if you want to make further changes.

However, normally neither of these restrictions are too inconvenient and most of the time you will not even notice. Rest assured that the removal of the original message and the update are done in a transaction. So, if the operation fails for any reason the original message will still be on the queue and an error message will be displayed.

3.8. Creating a 'new' message

We have seen that the tool 'Empty' will create an empty message. This is useful if you wish to create a message from scratch. There is a similar tool, 'New', which will also reset the content of the message however it will look at the content of the message descriptor format field to determine what to make. This means, for example, that if you are currently editing a PCF message and press the 'New' tool then MQEdit will discard the message content except for a blank MQPCF header. Similarly if the format field contained “MQDEAD “ then a message containing just the Dead Letter Header would be created.

The 'New' tool is also very useful for creating user format messages. All you need do is change the format field to the structure you wish to create and then press the 'New' tool. MQEdit will scan the user format definitions and determine the first structure defined for the current Queue and Format values. Note that since the structure itself has no content at the time the 'New' tool is pressed the conditional values are not used to determine which structure to create. They will, of course, be used to determine which structure to display.

3.9. Message Context

The message context[6] is a set of fields in the message descriptor which identify aspects of the message such as the user or application which put the message. These fields are:

User Identifier	The User Identifier that put the message.
Accounting Token	Information that can be used to charge for the work done as a result of the message.
Application Identity Data	Application data related to the putting Application.
Put Application Type	The type of Application that put the message
Put Application Name	The name of the Application that put the message
Put Date	The date when the message was put in Coordinated Universal Time (UTC).
Put Time	The time when the message was put in Coordinated Universal Time (UTC).
Application Origin Data	Data related to the putting Application's origin.

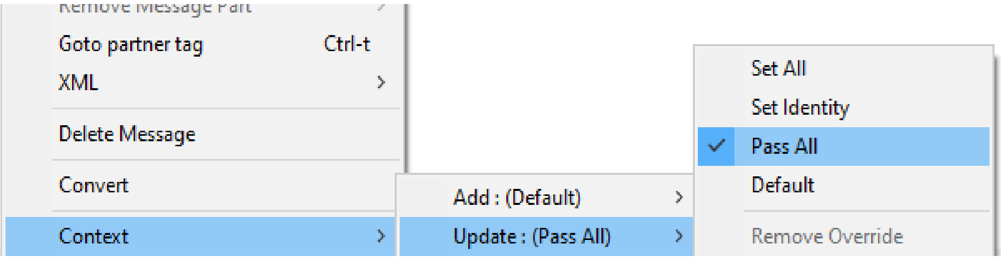
For an original 'new' message it is probably clear what values these fields should contain. They should contain values which reflect the putting application. However, if you are editing an existing message then it might not be quite so clear. For example, if you edit a message, should the user of the message reflect the user who made the edit or the user who originally put the message? The answer will depend on the exact situation, installation standards, and possibly even security settings. It is the IBM MQ message context options which control how these fields are treated by IBM MQ.

So, whenever MQEdit does a message operation it needs to know under what message context to perform the operation. To convey this to MQEdit, right-click on the message list window to open up the menu, choose the submenu called 'Context' and select the appropriate message context option.

The message context options, and a brief description of their effect, is given by the table below. These are IBM MQ concepts so more detailed information can be found in the IBM MQ documentation.

Set All	Allows the explicit setting of every context field in the message descriptor. In this mode all the fields in the Message Descriptor as taken 'as is'. Of course you need higher authority to use this option since you could impersonate other users. MQEdit will warn you if some of the context fields are empty
Set Identity	Allows the explicit setting of the identity fields in the message descriptor. These are the User Identifier, the Accounting Token and the Application Identity Data field.
Pass	The context fields are passed from the original message to the new message. This option is only available when copying, moving or updating a message.
Default	The context fields are set to the default values for this application. For example, the User Identifier will be set to the user running MQEdit.

In the case of the message edit window this 'Context' submenu is split into two further submenus – one for the 'Add' operation and one for the 'Update' operation. This recognises the fact that it may well be reasonable to have a different setting depending on whether you are creating a brand new message or editing an existing message. These two settings are initially set to the context values set in the “Preferences Dialog“ described on page 105. The two settings being “Initial Add Context“ and “Initial Update Context“ respectively.



---

[6] The English language is one of the most descriptive in the world, full of synonyms and different ways of saying the same thing. However, despite this we are still sometimes faced with the situation where the same word refers to multiple things. We are in this position with 'Context'. A context menu is something used in a GUI to refer to a menu which is constructed and displayed depending on the current 'context' (or state) of the application. For example in MQEdit the context menu is the menu which is shown when user right-clicks on a Dialog. However, in this section we are talking about 'Context' in the sense of IBM MQ Context fields. Where things get confusing is that we also have to talk about context menu options in the context menu! We will do our best to try and describe things in a way to minimise this confusion. In other words hopefully the meaning should be clear from the context!

Chapter 4. Data Conversion

Data conversion is one of those subjects that some people get very concerned about and others barely know exists. It generally depends on your MQ network and whether it spans formatting boundaries. By that I mean whether you have a mixture of ASCII and EBCDIC machines and/or a mixture of big-endian and little-endian machines. If you don't and all your machines are the same format then it is entirely possible you haven't given data conversion much consideration. It is up to you as to whether you want to read this chapter. Sadly, for the rest of us, data conversion is a necessary evil.

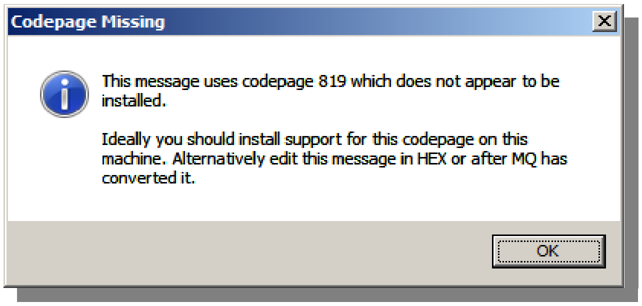
MQ messages are self describing in terms of data formatting. The Message Descriptors specifies what codepage and encoding the message data is in. Subsequent structure fields may alter that choice but it is always clear what codepage a character field is in and what encoding a numeric field is in. Generally speaking in MQ the recommendation is to use 'receiver makes good'. What this means is that you send out the message in your local format and anyone who needs to process the message should convert it in to their own format if required. This tends to result in the least number of conversions and is therefore more efficient. For example, a message sent from a Windows machine, through a z/OS hub and down to another Windows machine might never actually be converted at all. This is good from an efficiency point of view since data conversion clearly will use CPU cycles; but also from a data point of view since there is no absolute guarantee that converting between two code pages will be able to convert every character from one codepage to the other. This is because, almost by definition, not each code-point is in each codepage.

So, which is it for the editor? Do we 'make good' the message when we download it to the Windows machine[7] or do we try and avoid conversion at all costs? Well, to some extent it is your choice however the recommendation is that messages are not converted and kept in their original formatting. In the editor context menu there is a 'Convert' option. If this option is checked then any message retrieved from the Queue Manager will be converted to the local encoding before being displayed. It is recommended that you leave this option unchecked. The reason is to try and keep the message as close to the original message as possible.

You may think this causes the editor a problem; it means it now has to try and display EBCDIC characters on a Windows machine which is clearly an ASCII machine. Can it do this ? Yes, effectively it can. As long as Windows knows the codepage involved and you are using a font which has those characters then the characters will display just fine. So, with MQEdit it is perfectly possibly to edit an EBCDIC message directly just by typing characters on the keyboard. MQEdit will convert the characters you type in to the codepage of the field you are currently editing. Of course it is possible that you type a character that is not supported in the target codepage in which case MQEdit will display a message explaining that. Ultimately, of course, you can always go in to HEX mode where you can enter any hexadecimal value you wish.

4.1. Codepage Support

In order to display data in non-native codepages it is necessary for MQEdit to convert the data from it's current codepage into UTF16 for display. To do this MQEdit calls through to the local operating system for conversion. It follows therefore you need to ensure that the relevant codepages are installed on you system. If MQEdit tries to display a message which contains codepages which are not installed it will display an error box titled 'Codepage Missing' as shown on the right. At this point it is recommended that the user installs the relevant codepage onto their system. If that is not possible then the message can still be edited in HEX mode or you could switch on the Convert option to have the messages converted to the local codepage before editing. Bear in mind though that this means that if the message is put back on the queue the codepage of all its string data will have changed.



This message uses codepage 819 which does not appear to be installed.

Ideally you should install support for this codepage on this machine. Alternatively edit this message in HEX or after MQ has converted it.

4.2. Displaying non-native codepages

Displaying characters in different codepages requires two things; first Windows needs to recognise the codepage and be able to convert them to UTF16 ('wide characters'). These UTF-16 characters can then be drawn on the screen, provided that you are using a font which knows what the character should look like.

The editor allows you set whatever font is installed to display your message in. Note however that some fonts have many more defined characters than others. If you are only going to try and edit messages in a generic codepage such as 850 then you can probably use just about any font you like. However, if you want to edit lots of different codepages then you will need a font which has a wide range of characters. By default the editor will try to use a font called 'DejaVu Sans Mono'. This font has a pretty good character coverage and looks quite good. However, it certainly doesn't contain all possible characters. If you wanted a greater range you would be better to use a Unicode font, something like 'Lucida Sans Unicode'. Of course if you have your favourite then by all means use that.

---

[7] Essentially convert the message to little-endian ASCII

4.3. Proportional Fonts

One thing you may have noticed is that neither DejaVu nor Lucida are non-proportional fonts. And yet the editor is clearly displayed in a non-proportional way. The reason is that the editor can simulate a proportional font with a proportional font. However, the simulation is not without it's issues. The designers of non-proportional fonts try very hard to make equally spaced characters look balanced even though they are different widths. Designers of proportional fonts on the other hand make no attempt to make a 'W' look good when printed in the same space as an 'i'. If a character is wider it just takes up more space. The consequence of this is that displaying a proportional font in an equal width grid can look very gappy with some fonts. The problem is exacerbated still further if the font contains characters which are even wider than a 'W'.

It might pay to try out a few fonts and see which works best for you.

If you just need the invariant characters then the best looking display may well be with a non-proportional font. Good choices would be:

- Courier
- Courier New
- Consolas
- Lucida Console
- Lucida Sans

If you need a wide range of characters then good font choices would be:

- DejaVu Sans Mono
- Lucida Sans Unicode
- Arial Unicode MS
- Arial
- Caslon
- FixedSys
- Microsoft Sans Serif
- Tahoma
- Times New Roman

Each of these fonts have a different number of defined characters and so may be better for some codepages than others.

Of course you may have your personal preference especially if you regularly dealing with different codepages. Just remember that in order for MQEdit to display a character correctly Windows must know of the codepage and your selected font must be contain the desired character.

4.4. Bit endianess

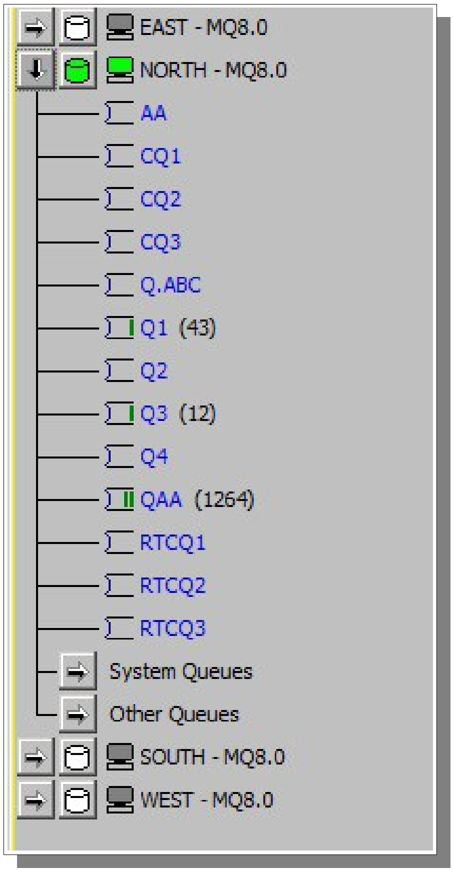
As we know Windows is an ASCII based, little-endian (reversed) machine. However, MQEdit still allows you to just type characters in to an EBCDIC field. It does this by converting the characters 'on the fly' for you. In the same manner you can just enter numbers into a big-endian (normal) field. MQEdit will byte swap the bytes if required.

4.5. Formatting defaults

If you press the 'New' tool to re-initialise the Message Descriptor what codepage and encoding should the editor use? Generally speaking you will probably want to use defaults related to the target Queue Manager. For example, if the Queue Manager is on a Windows machine then you might want codepage 850 and little-endian (reversed) encoding. However, if the Queue Manager is on a z/OS machine then you might want codepage 500 and big-endian (normal) encoding. When you first connect to a Queue Manager the editor selects the default values however you can override these choices in the Options tab in the location dialog if you wish.

Chapter 5. Queue Manager Pane

The Queue Manager pane is on the left-hand side of the main window. This pane shows you what locations (Queue Managers) you have defined and will show the list of available queues. Each Queue Manager entry consists of a number of pieces.



- **Twisty**
  Clicking on this twisty will alternatively show and hide the list of queues.
- **Queue Cache Icon**
  This icon will show whether MQEdit has cached the list of queues from the Queue Manager or not. Clicking on the icon will tell MQEdit to refresh the list.
- **Queue Manager Icon**

  An icon which show the current connection state of the Queue Manager.
  - Grey monitor indicates not currently connected
  - Green monitor indicates currently connected
  - Red cross indicates that connection attempt failed
- **Queue Manager Name**
  Clicking on the Queue Manager name will show a dialog of the Queue Manager definition, connecting to the Queue Manager if required.
- **Queue Manager Version (if selected)**
  If required you can choose to have the version of IBM MQ that the Queue Manager is running displayed after it's name. This is selected by clicking the *View-Show MQ Version* in the main menu.
- **Queue List**
  When the twisty is expanded MQEdit will show the list of queues defined on that Queue Manager. If you want you can choose to have the current depth of the queue displayed after the queue name. This is selected by clicking the *View-Show Queue* Depth in the main menu. The queues are defined into three separate lists.
  - **Local Queues**
    Since message can only exist on local queues and the editor is all about editing message it stands to reason that it is mainly the local queues a user is interested in so these are presented first.
  - **System Queues**
    It should not be necessary, and indeed could be hazardous, to change many of the system queues. A few of them do contain user messages such as SYSTEM.DEAD.LETTER.QUEUE and SYSTEM.CLUSTER.TRANSMIT.QUEUE so they are included here**.** However, if you choose to edit these messages then clearly care should be taken.
  - **All other Queues**
    None of the other queues contain message so they can't be edited. However, they could be targets for a drag-drop operation when moving/copying message so they are shown here.

5.1. Actions

The following actions are available in the Queue Manager pane.

5.1.1. Connecting to a Queue Manager

Clicking on the Queue Cache icon, opening the twisty or clicking on the Queue Manager name will all make an attempt to connect to the Queue Manager. If the connection succeeds the Queue Manager icon will turn green. If it fails the icon will change to a red cross. The location dialog for the given Queue Manager will display a message given the failing reason code. Follow normal IBM MQ practices, such as looking at the MQ error logs, to determine the cause of the failure.

5.1.2. Browsing a queue

Double clicking on a queue will browse that queue and display the messages in the message list pane.

5.2. Context Menu

By right clicking on the navigation pane a context menu is shown which allows you to do various actions against the selected Queue Manager.

5.2.1. Open Location...

Selecting this menu will bring up the location dialog for the selected Queue Manager.

5.2.2. Queue Manager

Selecting this menu will bring up the Queue Manager dialog for the selected Queue Manager.

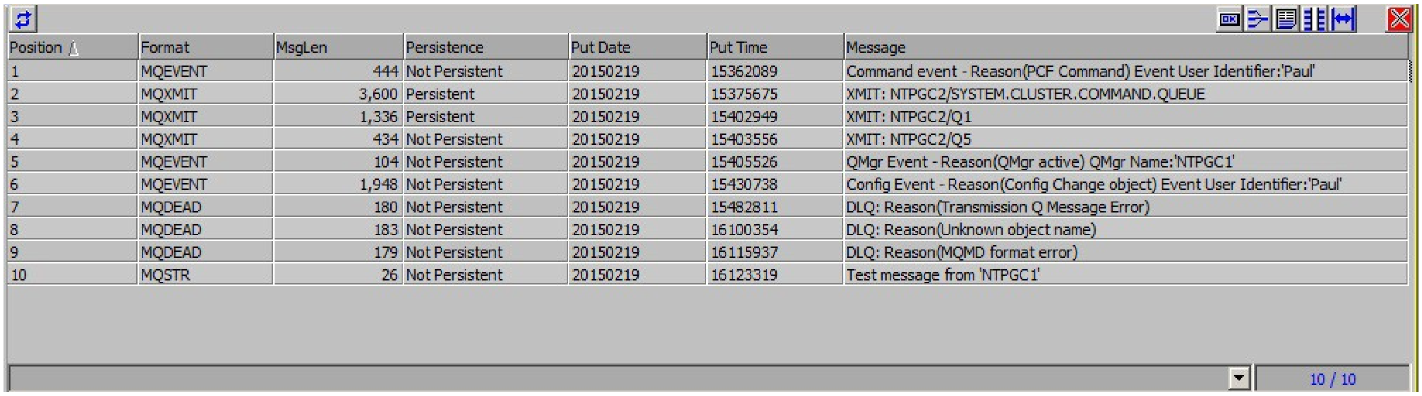
5.2.3. Show Queue Depth

Selecting this menu will alternately show and hide the queue depth counts shown against the local queue names. This setting applies to the whole pane, not just the selected Queue Manager. Note that the depths shown are the values that were in effect the last time the queue was queried. For an active queue this could be very different from the actual depth at the time. Pressing the queue cache icon will refresh the list of queues and their depths.

5.2.4. Disconnect

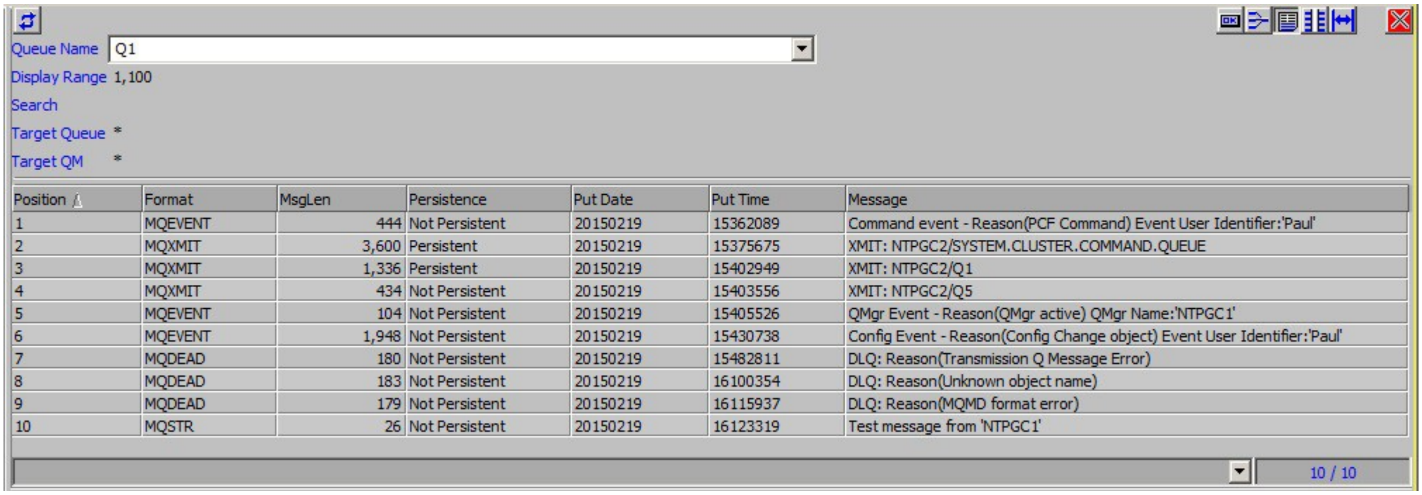
Selecting this menu will cause MQEdit to disconnect from the selected Queue Manager.

Chapter 6. Message List Pane



Position	Format	MsgLen	Persistence	Put Date	Put Time	Message
1	MQEVENT	444	Not Persistent	20150219	15362089	Command event - Reason(PCF Command) Event User Identifier:'Paul'
2	MQXMIT	3,600	Persistent	20150219	15375675	XMIT: NTPGC2/SYSTEM.CLUSTER.COMMAND.QUEUE
3	MQXMIT	1,336	Persistent	20150219	15402949	XMIT: NTPGC2/Q1
4	MQXMIT	434	Not Persistent	20150219	15403556	XMIT: NTPGC2/Q5
5	MQEVENT	104	Not Persistent	20150219	15405526	QMgr Event - Reason(QMgr active) QMgr Name:'NTPGC1'
6	MQEVENT	1,948	Not Persistent	20150219	15430738	Config Event - Reason(Config Change object) Event User Identifier:'Paul'
7	MQDEAD	180	Not Persistent	20150219	15482811	DLQ: Reason(Transmission Q Message Error)
8	MQDEAD	183	Not Persistent	20150219	16100354	DLQ: Reason(Unknown object name)
9	MQDEAD	179	Not Persistent	20150219	16115937	DLQ: Reason(MQMD format error)
10	MQSTR	26	Not Persistent	20150219	16123319	Test message from 'NTPGC1'

This pane shows the list of messages on the currently selected queue. The fields which are displayed can be selected in the Alter list dialog. The pane can also contains some configuration fields which can be displayed by pressing the 'Show Fields' menu/tool. In which case it would look like this:



Queue Name Q1
Display Range 1,100
Search
Target Queue *
Target QM *

Position	Format	MsgLen	Persistence	Put Date	Put Time	Message
1	MQEVENT	444	Not Persistent	20150219	15362089	Command event - Reason(PCF Command) Event User Identifier:'Paul'
2	MQXMIT	3,600	Persistent	20150219	15375675	XMIT: NTPGC2/SYSTEM.CLUSTER.COMMAND.QUEUE
3	MQXMIT	1,336	Persistent	20150219	15402949	XMIT: NTPGC2/Q1
4	MQXMIT	434	Not Persistent	20150219	15403556	XMIT: NTPGC2/Q5
5	MQEVENT	104	Not Persistent	20150219	15405526	QMgr Event - Reason(QMgr active) QMgr Name:'NTPGC1'
6	MQEVENT	1,948	Not Persistent	20150219	15430738	Config Event - Reason(Config Change object) Event User Identifier:'Paul'
7	MQDEAD	180	Not Persistent	20150219	15482811	DLQ: Reason(Transmission Q Message Error)
8	MQDEAD	183	Not Persistent	20150219	16100354	DLQ: Reason(Unknown object name)
9	MQDEAD	179	Not Persistent	20150219	16115937	DLQ: Reason(MQMD format error)
10	MQSTR	26	Not Persistent	20150219	16123319	Test message from 'NTPGC1'

These extra fields are usually not required and so are hidden but are used in some of the actions described below.

For efficiency, by default only the first few messages on the queue are displayed. This default can be changed in the Preferences Dialog, alternatively you can change the range for just this instance by modifying the 'Display Range' above.

6.1. Actions

The message list pane has a fairly extensive number of actions that you can request. All of these are available in the context menu and some of them are also available as tools in the toolbar or as buttons. We shall describe them from the point of view of the context menu:

6.1.1. Options

The 'Options' sub-menu contains the following:

6.1.1.1. Alter List....

This menu option allow you to specify what fields you would like displayed in the message list. For a description of the dialog please see 'Alter list dialog' on page 49.

6.1.1.2. Split List...

If required the list window can be split into two separate panes by selecting the 'split list' option from the context menu or clicking the ‘Split List’ tool in the toolbar. For example, the queue position can be kept in view in the left hand pane while the message attributes are scrolled in the right hand pane.

6.1.1.3. Reset Column Widths
Selecting this option will cancel any user defined column width and the system will choose what it considers to be the best width for the column, The user can set the required column width by dragging the edge of the column title.

6.1.1.4. Make Filter Default
Message lists can be filtered via a Boolean expression (see "Filtering" on page 54) to restrict which messages are shown. Selecting this menu item will associate the current filter with this list type. Each time the list is displayed it will have this filter. Selecting this menu with the filter window empty will disassociate any previous default filter.

6.1.1.5. Auto Refresh...
This displays a dialog which allows you to set an auto-refresh time interval and optionally ask for the returned information to be exported to a file. For more information please see "Auto Refresh Dialog" on page 47.

6.1.1.6. Copy to Clipboard
This option provides a quick way to copy some of the list data to the clipboard. The data which is copied depends on the state of the control keys at the time the option is selected.

SHIFT key pressed ?	CTRL key pressed ?	Action
No	No	Copy all fields of selected entries
Yes	No	Copy all fields of all entries
No	Yes	Copy the Message Token of the selected entries
Yes	Yes	Copy the Message Tokens of all entries

6.1.1.7. Export...
This option allows the user to export the displayed data to a file or the clipboard. When pressed a dialog will be displayed, please see "Export Dialog" on page 50 for a description of this feature.

6.1.1.8. Show/Hide Buttons
This option allows the user to Hide/Show the buttons of the dialog to make best use of the available screen area.

6.1.1.9. Show/Hide Toolbar
This option allows the user to Hide/Show the toolbar. The toolbar allows quick selection of common operations.

6.1.1.10. Show/Hide Fields
This option allows the user to Hide/Show the fields of the dialog to make best use of the available screen area.

6.1.1.11. Show/Hide Filter
This option allows the user to Hide/Show the filter windows of the dialog to make best use of the available screen area.

Please see " Filtering" on page 54 for a description of filtering expressions.

6.1.1.12. Show/Hide Column Filters
This option allows the user to Hide/Show the column filter windows displayed at the top of each column.

Please see "Column Filters" on page 54 for a description of column filters.

6.1.2. List Titles
The user can choose what values should be shown at the top of the field columns.

Option	Displayed value
Auto	MQEdit will automatically switch between 'Full' and 'Short' depending on how much space is available.
Full	A text description of the field
Short	A short text description of the field.
Field Name	The name of the field. These are the names that should be used in filters.

By clicking on the actual lists titles the list will be sorted by that column. Clicking in the title again will alternatively sort in ascending and then descending order. The list actually supports a primary and secondary sort field. You can select a secondary sort field by holding the 'Alt' key down while clicking on the field.

The edges of each column can be dragged via the mouse to re-size the columns. The user specified widths can be cancelled by selecting the ‘Reset Column Widths’’ pop-up menu option or by pressing the reset width toolbar icon.

The list of attributes displayed can be chosen from the ‘Alter list dialog“ menu option.

6.1.3. Refresh

This menu will refresh the current list of messages.

6.1.4. Open...

This menu will cause the currently selected message to be retrieved and displayed in the Message Edit pane. Note that not necessarily all of the message will be retrieved. The amount of data is set in the 'Browse' tab of the “Preferences Dialog“.

6.1.5. Detail...

Pressing this menu will bring up a new dialog showing the currently selected message.

6.1.6. Copy

Pressing this menu will copy the selected message to the queue identified by the Target Queue and Target Queue Manager fields.

6.1.7. Move

Pressing this menu will move the selected message to the queue identified by the Target Queue and Target Queue Manager fields.

6.1.8. Delete

Pressing this menu will delete the selected messages form the queue.

6.1.9. Put Message....

Pressing this menu will present a put message dialog which allows the user to put a simple text message to a queue. For more information please see “Put Message Dialog“ described on page 52.

6.1.10. Load/Unload Messages...

Pressing this menu will present a dialog which allows you to either load messages from a file on to a queue or unload messages from a queue to a file. Please see “Queue load/unload facility“ on page 91 for more information.

6.1.11. Convert

This menu option allows you to choose whether messages are converted to the local codepage and encoding before being displayed in the list. Generally speaking it should not be necessary to convert the messages and it is recommended that you don't convert messages when editing. However, the option is here in case the data is not understandable without first converting.

6.1.12. Message Summary

This menu option controls whether the message is shown just as the first few bytes of the message or whether MQEdit should try to summarize the contents of the message. MQEdit understands MQ message formats such as event messages, PCF messages, dead letter queue headers and transmission queue headers. For 'User Format Messages' you can choose which fields are shown in the summary. The summary will only be based on the first few bytes of the message. The number of bytes can be controlled, to some extent, by settings in the “Preferences Dialog”.

6.1.13. Strip Headers

Messages on an IBM MQ queue may have a transmission queue or dead letter queue header prepended to the actual application message. By default it is assumed that these headers should be maintained if the message is moved or copied to another queue. This helps to ensure that the full message is maintained. However, there are times when these headers should be stripped off, for example, when moving a message from a transmission queue or Dead Letter Queue to a new target.

6.1.14. Multi Transaction

Normally all copy, move and delete operations will be done under one transaction, this avoids the problem that a system crash or some sort of failure will leave the operation only partially completed. All the operations will either complete or the transaction will be backed out. There is a queue manager attribute, max uncommitted messages, which dictates the maximum number of messaging operations allowable by an application in a single transaction. If you are trying to move or copy more messages than this limit then clearly it can not be done in a single transaction. In these cases you must select the ‘Multi Transaction’ option and take the small risk that the operation may only partially complete.

6.1.15. Search Offset

When using a search string in a message list display this setting controls whether the 100 byte message portion that is displayed is from the start of the message or from where the search string is found within the message.

6.1.16. Message Operations

6.1.16.1. Apply to all messages

Selecting this option will change the Move, Copy and Delete, operations to apply to all messages on the queue. The menu text will be changed to reflect this state. Because this is a potentially hazardous operation the ‘All message’ mode is removed as soon as any action is selected.

6.1.16.2. Move/Copy messages directly to remote QMgrs

MQEdit supports moving and copying messages between Queue Managers. There are two ways this can be done. Either it can be done indirectly through MQ channels and transmission queues. Or it can be directly by MQEdit having a connection to each Queue Manager and using the connections alternately.

Using the direct approach is much faster and simpler. When copying messages there is no risk in using the direct method. However, when moving messages there is a small risk that a failure will occur between committing the messages at the target Queue Manager and then committing at the source Queue Manager. The consequence of this is that in very rare circumstances the messages could be duplicated. One thing you can do to mitigate the situation is to move the messages to a holding queue on the target Queue Manager. If the move operation is successful then you can move the messages to the actual target Queue. This second move would be within a single Queue Manager and can therefore be done in a single transaction.

6.2. Drag and Drop Support

Messages can be moved or copied using normal drag&drop actions with the mouse. The procedure is as follows:

- Select the message or messages you want to move/copy in the message list
- Hold the mouse over any of the messages and press and hold mouse button one
- Drag the mouse over to a target queue in the navigation window
  This queue can belong to any Queue Manager and in addition to local queues, the target may also be a remote or alias queue. The mouse pointer will change to let you know the target queue and whether you are copying or moving. By default you will be doing a move operation but holding the <SHIFT> button will change the operation to copy.
- When you are happy with your target let go of the mouse button

You can drag&drop messages directly between Queue Managers depending on the menu setting of “Move/Copy messages directly to remote QMgrs“.

6.3. Fields

By default the Message List pane is shown without showing the fields since they are only required for specialised operations. However, there are times when they are very useful. The fields can be displayed by selecting the 'Show/Hide Fields' menu action.

6.3.1. Queue Name

This field indicates which queue is being browsed. Normally you would set this field by double-clicking on a queue on the queue manager navigation pane. However, if you want you can change the queue name directly, either by typing the name or selecting from the drop-down, and then pressing the refresh button will show the messages on that queue.

6.3.2. Display Range

When a list of messages is requested you have the option to specify the range of messages you’re interested in. By default the range is ‘1,100’, which means only the first 100 messages. This prevents accidentally displaying thousands of messages if the queue contains more messages than you expected !

However, if you find that you frequently want to browse a different number of messages than the first 100 you can change this setting in the preferences dialog. See “Preferences Dialog“ on page 105 for more details. MQEdit will support browsing up to 30,000 messages in a single display. Again you can set this maximum in the “Preferences Dialog“. If you really need to browse a message that is not in the first 30,000 messages on a queue then you can use the display range to specify the starting point. However, a much more efficient way of doing it is to use the selector field below.

6.3.3. Selector

The selector[8] field allows the user to enter an SQL92 expression which is executed at the server. The Selector field allows you to request that only certain messages, matching a particular criteria, are browsed from the queue. If you have a deep queue then clearly this is a much more efficient way of looking for particular messages. MQEdit also allows additional filtering to be done in MQEdit itself, see “Filtering“ on page 54 for more information.

Descriptions of the format of the expression can be found in the Message Selector Syntax section of the IBM MQ documentation.

Simple examples would be:

- **Root.MQMD.Persistence=1**
  Show only the persistent messages
- **Root.MQMD.UserIdentifier = 'clarkep '**
  Show only messages put by a particular user
- **Root.MQMD.MsgId="0x414D51204E5450474331202020202020B51261522004F009"**
  Select a particular message
- **myprop > 10**
  Show only messages where the value of property ‘myprop’ has a value greater than 10.

Note that all field names in selectors are case sensitive.

6.3.4. Search

The search field allows the user to enter a search string for the messages. Only messages which contain this case sensitive search string will be displayed. Note that only the message itself, not the message descriptor is checked for the string. When a search string is used the display range fields control which of the matching messages are displayed. For example, 3,5 will display the 4th, 5th and 6th matching messages. Note that the ‘Search Offset’ setting controls whether the portion of the message shown starts at the point where the search string is found or whether it is the start of the message. If the search offset is used the message summary is automatically inhibited since the bytes returned from the command server are no longer at the start of the message.

6.3.5. Target Queue

When copying or moving messages to another queue you can optionally identify the target queue for the message, by default it has the value ‘*’ which has the following effect:

- If the message has a MQXQH or MQDLH header then the queue manager is that which is specified in the header provided the option 'Strip Headers' is selected.
- If the message doesn’t have a header then it is the local queue manager

6.3.6. Target QM

When copying or moving messages to another queue you can optionally identify the target queue manager for the message, by default it has the value ‘*’ which has the following effect:

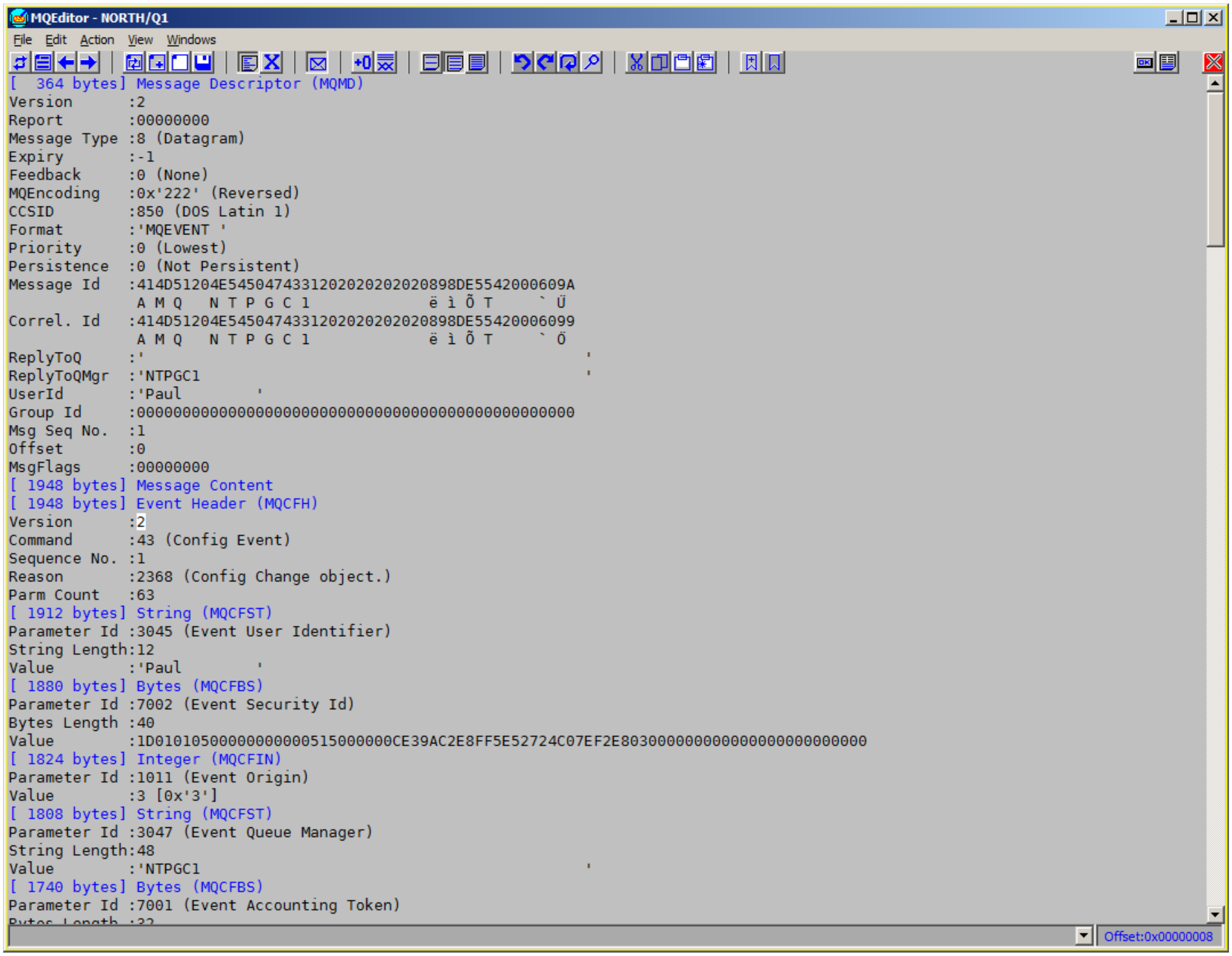
- If the message has a MQXQH or MQDLH header then the queue manager is that which is specified in the header provided the option 'Strip Headers' is selected.
- If the message doesn’t have a header then it is the local queue manager

---

[8] Available on Queue Managers MQ V7.0 and later

Chapter 7. Message Edit Pane

This pane is where MQEdit actually displays the message and where you will do all of the editing. To give you more space you can double-click on the pane and it will expand to fill the whole window area. Double clicking again will revert back to the 3-pane view.



```
[  364 bytes] Message Descriptor (MQMD)
Version      :2
Report       :00000000
Message Type :8 (Datagram)
Expiry       :-1
Feedback     :0 (None)
MQEncoding   :0x'222' (Reversed)
CCSID        :850 (DOS Latin 1)
Format       :'MQEVENT '
Priority     :0 (Lowest)
Persistence  :0 (Not Persistent)
Message Id   :414D51204E5450474331202020202020898DE5542000609A
              A M Q   N T P G C 1             ë ì Õ T     ` Ü
Correl. Id   :414D51204E5450474331202020202020898DE55420006099
              A M Q   N T P G C 1             ë ì Õ T     ` Ö
ReplyToQ     :'                                                '
ReplyToQMgr  :'NTPGC1                                          '
UserId       :'Paul        '
Group Id     :000000000000000000000000000000000000000000000000
Msg Seq No.  :1
Offset       :0
MsgFlags     :00000000
[ 1948 bytes] Message Content
[ 1948 bytes] Event Header (MQCFH)
Version      :2
Command      :43 (Config Event)
Sequence No. :1
Reason       :2368 (Config Change object.)
Parm Count   :63
[ 1912 bytes] String (MQCFST)
Parameter Id :3045 (Event User Identifier)
String Length:12
Value        :'Paul        '
[ 1880 bytes] Bytes (MQCFBS)
Parameter Id :7002 (Event Security Id)
Bytes Length :40
Value        :1D010105000000000005150000000CE39AC2E8FF5E52724C07EF2E803000000000000000000000000
[ 1824 bytes] Integer (MQCFIN)
Parameter Id :1011 (Event Origin)
Value        :3 [0x'3']
[ 1808 bytes] String (MQCFST)
Parameter Id :3047 (Event Queue Manager)
String Length:48
Value        :'NTPGC1                                          '
[ 1740 bytes] Bytes (MQCFBS)
Parameter Id :7001 (Event Accounting Token)
Bytes Length :32
```

The editor has many features which we will talk about soon. Let's just take a minute to familiarise ourselves with the layout. The toolbar along the top contains a number of the common operations. All these operations, and more, are available in the context menu. However, the toolbar is generally more convenient and provides useful status of the options in force and the status of the message.

For example, when you load a message with the intention of editing it it may be a good habit to look at the second icon in the toolbar. This tool will show whether the entire message has been retrieved from the server. Remember that in the “Preferences Dialog“ you can specify how much of the message should be retrieved by default. This is to prevent you from loading an 50MB message by accident. If the message itself is larger than the default retrieve size then MQEdit will display the message incomplete icon ( ). This icon, by it's jagged edge bottom, indicates that not all of the message is present. By pressing this icon MQEdit will go and retrieve the entire message and, providing this is successful, the message complete message icon ( ).

Below the toolbar is the actual edit area. This area contains both the message descriptor and the message content. The cursor can move between these two areas as though they were in contiguous storage although clearly they never are. This can be useful how ever if you want to copy/paste the data another document.

7.1. Cursor

The cursor can be moved around the screen in the normal manner using the cursor keys or the <TAB> key. Similarly clicking on the pane with the mouse will move the cursor to an appropriate position. If the line contains an editable field then that field with be shown in a different colour[9]. The cursor can be displayed in different ways:

Cursor Shape	Meaning
Vertical Line	Cursor is on an editable field in insert mode Pressing the <INSERT> key will toggle between insert and over-write mode
Half Block	Cursor is on an editable field in over-write mode Pressing the <INSERT> key will toggle between insert and over-write mode
Horizontal Line	Cursor is not on an editable field

Normal key combinations are supported for manipulating the cursor. These are:

Keys	Meaning
<UP>	Move cursor up one line if not at the beginning
<DOWN>	Move the cursor down one line if not at the end
<LEFT>	Move left one character unless at the beginning of the line in which case move to the end of the previous line.
<RIGHT>	Move one character to the right unless at the end of the line in which case move to the start of the next line
<END>	Move to the end of the field or line depending on line type
<HOME>	Move to the start of the field or line depending on the line type
<CTRL> + <HOME>	Move to the start of the data
<CTRL> + <END>	Move to the end of the data
<Page Up>	Move up one page of data
<Page Down>	Move down one page of data

7.1.1. Selecting Data

Selecting the data can be done by holding down the <SHIFT> key while using any of the key combinations above or by swiping the mouse with mouse button 1 held down. Selecting data is useful for copy/paste and deleting large number of characters.

---

[9] The exact colours used can be changed so it is possible to removing this highlighting by choosing the same colour for both.

7.2. Editing a field

Each output line either contains one or zero editable fields. If the cursor is on an editable line then the editable field is highlighted in a different colour.

Changing the value of a field can be done in a number of ways:

- **Over writing the value can be done just by typing over the field characters**
  Note that not all characters can by typed in to all fields. For example you can not type the character 'M' in to a numeric field.
- **Inserting new characters can by done by typing in insert mode**
  Some fields are fixed length however so inserting characters will involve discarding value off the end of the field. MQEdit will not allow you to discard data of value. Therefore, for fixed length string fields there must be enough trailing spaces to cover the data you are trying to insert. For fixed length hex fields the field must have trailing zero characters. There are other rules if editing hex, for more information refer to “Editing Hex“ on page 41.
- **Activating selection list**
  Some fields have a known set of possible values. For these fields you can activate the selection list display by clicking on the small box to the right of the field (or pressing <F4>)
- **Pasting in data from the clipboard**
  Pasting is always an insert operation. If you want to over-write the data then select the same number of characters as you are about to paste.

The editor is a 'live parsing' editor which means that as changes are made the editor will reflect the changes not just in the field being changed but on the whole display. For example, consider the display of a simple string message. If you were to change the FORMAT field in the Message Descriptor you radically change how the message is interpreted. For example instead of displaying a string of characters for the message content the editor will most likely display the message as hex characters. This live parsing can greatly help in seeing the effect of the edits and confirming that the changes are correct. However, there are some fields which are handled slightly differently which we shall talk about now.

7.2.1. Formatted fields

Some fields can only contain characters in a particular format. Consider, for example, the lowly integer. Clearly a number field can only contain certain characters (for example '0' – '9' & '-') but those characters can only occur in certain orders. If you were to type in '-1-1-1-1' as a number it would clearly be nonsense[10]. After all this isn't an expression you are typing in but a single number. So, for these types of fields the editor will check whether the format of the field is correct. If it isn't the the field will be displayed in a different colour and the change **will not** be applied to the actual message data. If the cursor is moved to another field then the data in the field will revert to the last valid value.

7.2.2. Deferred Fields

There are some fields that applying the values immediately may have unwanted effects. Consider, for example, a codepage value. Suppose you have a message with codepage of 850 and you would like to change it to 437. It is not possible (without using copy/paste) to change the number 850 to 437 without going through some intermediate numbers which you wouldn't want. You don't want, for example, the editor trying to display the message in codepage 437850!

With deferred fields the field edit is displayed on the screen but is only applied to the data when you press the <ENTER> key. If you want to cancel the edit then just move the cursor to another field.

7.2.3. Expert Fields

One of the powerful features of MQEdit is it's ability to modify 'other' fields when you make changes. For example, consider an MQCFST structure which is part of the PCF message. This is a fairly simple structure but it would be very awkward to edit if you had to change all the fields. The reason is that the structure has two length fields, one of which is rounded up to the next fullword. Life is much easier if you let MQEdit update these values for you. If you try to edit these fields you will be shown a message which says you must be in 'Expert Mode' to edit the field.

To go into 'Expert Mode' is very easy, just select it from the 'Edit' main menu. However, be careful in this mode since it is very easy to make edits which will destroy the structure of your message.

---

[10] In the case of integers the editor will also not allow leading zeroes. Although the number '00000123' is the same numerically as '123' the zeroes do not add anything to the meaning. The editor therefore treats these as a typo and forces you to remove any leading zeroes.

The main purpose of 'Expert Mode' is to repair messages. It is possible that you have a message which already has incorrect values for some of these length fields. By switching to 'Export Mode' and entering the correct value you will immediately see structures which follow displayed correctly.

7.2.4. Adding and Removing Structures

One of the powerful features of MQEdit is that it understands not only the standard MQ structures but how they go together. So, whereas constructing these messages by hand can be difficult, laborious and error prone using MQEdit is becomes very easy. Essentially there are two top level menu actions:

7.2.4.1. Add Message Part

Ensure that the cursor is at the point where you would like to add the new message part. Depending on the context of the cursor you will be offered a choice of the following:

- Add Message Header Before
- Add Message Header After
- Add Message Structure Before
- Add Message Structure After
- Add Array Element

7.2.4.2. Remove Message Part

Ensure that the cursor is at the point where you would like to delete the message part. Depending on the context of the cursor you will be offered a choice of the following:

- Remove Message Header
- Remove Message Structure
- Remove Array Element

7.2.5. Repeat Last Action

The repeat last action menu is a useful short-hand for occasions where you need to do the same thing again and again. For example, consider the case where you need to add a few MQCFIN structures to a PCF message. All you need do is add one using the appropriate menu and then press the 'repeat last action' button as many times as required.

However it can be useful for a number of situations:

- Adding many structures of the same type
- Removing the current structure
- Adding new array elements
- Removing the current array element

7.3. Actions

The message edit pane has a fairly extensive number of actions that you can request. All of these are available in the context menu and some of them are also available as tools in the toolbar or as buttons. We shall describe them from the point of view of the context menu:

7.3.1. Options

7.3.1.1. Show/Hide Buttons

This option allows the user to Hide/Show the buttons of the dialog to make best use of the available screen area.

7.3.1.2. Show/Hide Toolbar

This option allows the user to Hide/Show the toolbar. The toolbar allows quick selection of common operations.

7.3.1.3. Show/Hide Fields

This option allows the user to Hide/Show the fields of the dialog to make best use of the available screen area.

7.3.2. Refresh

This menu allows you to refresh the currently selected message. If the current message has been changed then you will be asked for confirmation that you wish to discard the current edits.

7.3.3. Get Complete Message

By default MQEdit will only retrieve the first few bytes of a message. You can set how many bytes by changing the setting in the 'Browse' tab of the Preferences Dialog. If the message is larger than this amount then the message will still be shown but it will be incomplete (truncated). By selecting this action you can request that MQEdit retrieves the whole of the message. In order to edit the message you must have retrieved it all.

7.3.4. Prev

When browsing through a queue this action allows you to retrieve the previous message.

7.3.5. Next

When browsing through a queue this action allows you to retrieve the next message.

7.3.6. Update

This action requests that the currently selected message is 'updated' with the current contents of the edit buffer. This action is only available if you originally selected a message.

Updating a message is done by transactionally removing the old message and putting the new message to the queue. This ensures that there can never be two copies of the message on the queue. However, it does not preserve message order. There is no way, currently, to do an 'in place' update of an MQ message.

The other consequence of the update is that the message is actually a new physical message. This means that if you wish to edit the message again then you will need to re-select the updated message.

7.3.7. Add

This action will take the contents of the edit buffer and add it to the queue as a new message.

7.3.8. New

This action will reset the contents of the message buffer and Message Descriptor. You use it when you want to 'start again' or generate a message from scratch.

7.3.9. Detail...

This action will show a dialog which details all of the message descriptor fields. The dialog also allows you to do many of the actions described in this manual.

7.3.10. Save Message...

This action will bring up a simple dialog which allows the user to specify a file into which the message can be saved. Messages are save in 'QLOAD' format so can be loaded in a variety of tools. This action will not affect the content of a queue in any way.

7.3.11. Load Message...

This action will bring up a simple dialog which allows the user to specify a file from which a message can be loaded. The file must be in 'QLOAD' format. Only the first message in the file will be loaded, any further messages will be ignored. If you wish to load a set of messages to a queue then this can be done using the 'Load/Unload Messages...' action in the message list display. This action will not affect the content of a queue in any way.

7.3.12. Formatter

MQEdit understands many formats of message and many of these it can auto-detect. This is done by a combination of looking at the Format field in the MQMD and the first few bytes of the message. For example, all of the standard MQ headers such as MQXQH, MQDLH and MQRFH2 can be determined by looking a the Format field. Whether the message is in XML requires looking at the first few bytes to see whether it starts with a '<' character. There are other formats which are not quite so easy to detect or that may be only desirable in certain circumstances. Using this menu selection you can choose how MQEdit should try to display the message. The choices are:

- **Auto**
  This is the default setting and allows MQEdit to choose the most appropriate way of displaying the data. Often Auto with be the correct setting but select a different formatter if you wish to see the data differently.
- **Plain Text**
  The data is displayed as just plain text characters.
- **XML**
  MQEdit will parse the data into the various XML tags.
- **JSON**
  The data should be treated as JSON data and formatted according to the bracket nesting.
- **Comma Separated Values**
  The data consists of many fields of data separated by commas. Each comms separated value is displayed on a new line. This format is also useful for displaying the Open Financial Standard (OFS) messages.
- **Edifact**
  The data should be interpreted according to the EDIFACT standard.
- **FIX**
  This formatter is designed for FIX (Financial Information eXchange) messages. Essentially these are message where the delimiter between fields is the SOH (0x01) character.

Depending on the Formatter in use the following options may also have an effect on the editing behaviour.

- **Contextual Newline**
  For some formats the action that MQEdit should perform when the user presses <ENTER> is switchable. If the option is set then MQEdit will do the following instead of entering a Newline character
  - **FIX**
    An SOH (0x01) character is inserted into the message.
  - **CSV**
    A comma is inserted into the message.

7.3.13. Display Format

Sub-menu containing a number of options controlling how you would like the data displayed.

7.3.13.1. Formatted Message

When this option is selected MQEdit will try to display the message in a formatted display. MQEdit understands many MQ formats such as PCF and Event Messages, Transmission Queues, Dead Letter Headers, RFH2 etc. However, it can't possibly understand all message formats[11]. If it detects a format it doesn't understand then it will display the data in hex.

---

[11] In future releases we hope to add the concept of user formats where the FORMAT field of the MQMD is used to select an appropriate structure. If you feel this would be useful to you or you have other formats you think should be implemented please get in touch. We are always interested to hear of new customer requirements.

7.3.13.2. Hex Message

It is always possible to select this option and display and edit the message in hex if required. The advantage of hex editing is that it is always available regardless of the message content and there are no limitations on what value you can put in what bytes. However, bear in mind that it can be very easy to create a message which is not structurally correct.

Editing in hex is fairly simple but you should read the section 'Editing Hex' on page 41 before you attempt it.

7.3.13.3. Message Descriptor

This option toggles whether the message descriptor is shown or not.

7.3.13.4. Display Offset

This option selects whether you would like the editor to display the lines with a column down the left-hand side giving the message offset of the field.

7.3.13.5. Display Hex as chars

When selected the editor will display the characters represented by a hex field below the hex string. Bear in mind that the characters shown are just a 'guess' since hexadecimal fields have no inherent codepage. The editor will guess whether the hex bytes are ASCII or EBCDIC. The characters can be edited as if they were a normal string field.

7.3.13.6. Character Substitution Mode

Sometimes messages can contain sequences of characters which represent a single character. For example, the sequence &gt; is sometimes used to represent the 'greater than' (>) symbol. Having these character sequences can be easier for a target program to process but can make the message difficult to read. By selecting this mode the user can display these sequences by the single character they represent. Note however that the message itself will still contain the original sequence. The substitutions performed are the following:

Sequence	Character
&gt;	**>**
&lt;	**<**
&nbsp;	**(space)**
&amp;	**&**
&quot;	**“**
&apos;	**'**

Editing of messages is still allowed in this mode. However, by default a confirmation message is shown to ensure that the user is aware that they are editing a message where what is displayed is not necessarily the content of the message. The danger with editing in this mode is that the reverse substitution is not made. For example inserting a '>' will not insert the sequence &gt;. One could easily end up with a message containing a mixture of &gt; and > characters if you are not careful.

7.3.13.7. Auto Detect XML Message

When selected the application will examine the first few bytes of the message to see whether it contains XML tags. If it does the message will be formatted as though it were an XML message. If not selected only messages with the MQRFH2 format will be treated as XML.

7.3.13.8. Display XML shortform

When selected, any XML message will be displayed in an abbreviated form where not all tags and tag characters are displayed to aid readability.

7.3.14. Display Level

7.3.14.1. Low

Low detail. Only the most important fields are shown.

7.3.14.2. Medium

Medium detail. Most fields are shown except for supporting fields such as eye-catchers, version numbers and lengths.

7.3.14.3. High

High detail. All fields are shown.

7.3.15. Display Line Number

By default MQEdit will display the offset, in bytes, of the cursor in the message position at the bottom right of the pane. However, if you prefer you can select this action and have the position displayed as a line number. Bear in mind that the line number can vary greatly depending on the message format display settings such as formatting, message detail and displaying hex as chars.

7.3.16. Undo

Every edit you make to the message can be undone. The undo change is unlimited but is discarded as soon as a new message is selected. The standard key sequence of Ctrl-z can be used.

7.3.17. Redo

This action allows you to redo any operation that you have previously redone. The standard key sequence of Ctrl-y can be used.

7.3.18. Repeat Last Action

Certain options, such as adding and removing message parts, can be repeated. This can make adding multiple array elements or multiple structures such as MQCFIN, much quicker.

7.3.19. Find and Replace

This action will display a Find and Replace dialog, please see the 'Find and Replace Dialog' description on page 45 for more information.

7.3.20. Clipboard

7.3.20.1. Select All, Cut, Copy, Paste, Paste All

The actions support the standard clipboard copy and paste functionality with some minor additions. Please read the section on 'Copy and Paste' on page 43 for the full story.

7.3.21. Bookmark

Particularly for large messages it can be useful to be able to remember your place in the message.

7.3.21.1. Save Bookmark

This action will remember the current cursor position in a message.

7.3.21.2. Goto Bookmark

This action will restore the cursor position to a previous saved bookmark.

7.3.22. Add Message Part

MQEdit understands a number of message formats. The actions below allow you to tell MQEdit to add certain message elements.

7.3.22.1. Add Message Header (Before/After)

The first part of any structure message tends to be a message header. There are a whole variety ranging from Dead Letter Headers, Transmission Queue headers to Programmable Command Format (PCF) Headers. Using these actions allow you to add a new header either before or after the current cursor position.

If required the header is 'chained' in to any existing chain to maintain the message structural integrity.

7.3.22.2. Add Message Structure (Before/After)

Structured messages, most notably Programmable Command Format (PCF) messages consist of a number of consecutive structures. These actions allow you add new structures either before or after the current cursor position.

7.3.22.3. Add Array Element

Some structures, such as MQCFIL and MQCFSL, represent arrays of data. This action allows you to simply add another element to the array.

7.3.23. Remove Message Part

The following actions are concerned with removing elements from the message.

7.3.23.1. Remove Header '<Header Name>'

By placing the cursor on a message header and selecting this action MQEdit will remove the message header and 'de-chain' it if required. Note that you cannot remove a message header if there are dependant following structures – if MQEdit allowed this the message structure integrity would be compromised.

7.3.23.2. Remove Structure '<Structure Name>'
Placing the cursor anywhere in a message structure and selecting this action will remove the entire structure from the message.

7.3.23.3. Remove array element
Placing the cursor on an individual array element and selecting this action will remove that particular array element.

7.3.23.4. Goto partner tag
The purpose of this action is to position the cursor on the matching element of the message. The behaviour is slightly different depending on whether the message is XML or not.

- **XML**
  With the cursor positioned anywhere in an XML tag, that includes on or after the start or end angle bracket, using this menu item, or its hot-key equivalent <Ctrl-t>, will reposition the cursor at the beginning of the partner tag.
- **Any other format**
  MQEdit will position the cursor on the matching bracket. The cursor must, naturally, be currently places on a bracket. Brackets of [], (), {} and <> are supported.

7.3.24. XML
The following actions are concerned with editing XML in a message.

7.3.24.1. Select XML
With the cursor positioned in an XML tag, either the start tag or the end tag, using this menu item will select all the text between the start and the end tag, excluding the tags themselves.

7.3.24.2. Select XML (incl tag)
With the cursor positioned in an XML tag, either the start tag or the end tag, using this menu item will select all the text between the start and the end tag, including the tags themselves.

7.3.24.3. Comment
With some text selected, using this menu item will surround the selected text with XML comment markers

```
<!-- -->
```

If no text is selected, using this menu item will insert an empty comment marker.

7.3.24.4. Uncomment
With the cursor positioned anywhere in an XML comment tag, that includes on or after the start or end angle bracket, using this menu item will remove the comment markers.

7.3.24.5. Add XML tag
With some text selected, using this menu item will surround the selected text with XML tag markers, and position the cursor at the place where the tag name should be typed. If XML Assist is enabled (see 19.1.5. XML Assist on page 105) then typing the start tag will automatically fill in the end tag at the same time.

With no text selected, using this menu item will insert XML tag markers at the cursor position, and position the cursor at the place where the tag name should be typed. With XML Assist enabled the same behaviour can be achieved by typing the open angle bracket.

7.3.24.6. Remove XML tag
With the cursor positioned in an XML tag, either the start tag or the end tag, using this menu item will remove both the start and end tag, leaving any text that was between the tags intact.

7.3.25. Delete Message
This action will delete the current selected message.

7.3.26. Convert
In general it is advised that you do not use a converted messages as the starting point for your edits. This is to ensure that the edited message is a similar as possible to the original message. However, there could be cases where you wish to convert the message.

7.3.27. Context

The context options control how much of the Message Descriptor is taken from the editor and how much of it is generated at MQPUT time. Different context options require different levels of authority with 'Set All' requiring the most. For more information about Put Message Context refer to the IBM MQ documentation.

7.3.27.1. Set All

This option states that the entire message descriptor should taken 'as-is'.

7.3.27.2. Set Identity

This option states that just the Identity Context fields should be taken from the edit Message Descriptor.

7.3.27.3. Pass

This option can only be used when updating a message. It states that the message context should be taken from the original message being edited.

7.3.27.4. Default

This option requires the least amount of authority. Messages are added to a queue with default context. This means that MQEdit will be the putting application and the user of MQEdit would be in the identity context.

7.4. Fields

By default the Message edit pane is shown without showing the fields since you don't normally need to see the values.

7.4.1. Position

This read only field gives an indication of the position of this message in the queue. Of course, the position was only a snap-shot of the position and it could have changed since the browse. Note that if you are using selector the position only reflects the relative position of this message on the queue not its absolute position if a selector had not been used.

7.4.2. Max Message Size

By default MQEdit will only retrieve the first few bytes of the message. This can be set in the 'Browse' of the 'Preferences Dialog'. However, if you wish to make a one-time change to the number of bytes you can update this field and press refresh. Note that if you wish to retrieve the entire message then it is simpler and easier to just press the 'Get Complete Message' action.

7.4.3. MsgLen

This field shows the size of the message as it was retrieved from the queue before any editing.

Chapter 8. Editing Hex

At first glance it might appear that editing hexadecimal digits would be straightforward and certainly if you are in over-write mode it is. However, life is more complicated if you are inserting data. An issue arises because the the user may be entering an odd number of nibbles[12] and yet messages must, clearly, be an even number of nibbles. Powerful as IBM MQ is you can't put a message that is half a byte long!

8.1. Nibble Insertion

Consider the following hexadecimal string:

```
123456
  ^
```

And we want to enter the nibble '7' at the given position. What should happen? Clearly there are two sensible choices, these are:

```
12734560                             * MQEdit does not do this
   ^
```

and

```
12703456                             * MQEdit does this
   ^
```

As we said before we must have an even number of nibbles so, despite the user only entering one nibble, we have to actually add two nibbles. So, have to add a padding nibble. The choice therefore is whether this padding is added after the insertion point or at the end of the hex string. Well, as is shown about MQEdit will add the padding immediately after the insertion point. The advantage of this is that rest of the data maintains the it's byte/character relationship. In other words the data moves forward in integral bytes and therefore nibbles that were the first nibble in a byte are still the first nibble in a byte.

So, after inserting this nibble we are now in this position:

```
12703456                             * MQEdit does this
   ^
```

Notice that the insertion point is now not pointing at the start of the byte but is instead pointing at the second nibble of the byte. So, suppose we insert a '8' at this point. Well, we get this.

```
12783456                             * MQEdit does this
    ^
```

Notice that this time we don't add a padding nibble. If we did that then the hexadecimal string would get too big. Adding six nibbles would increase the string by six bytes! So, the rule is that padding is only done if you are inserting at the first nibble of a byte. The rule for the second nibble is that insertion is only allowed if that nibble is zero. This is try and prevent accidentally writing over data when you are insert mode.

Now, what do we do for deletion?

---

[12] A nibble is one half of a byte or put another way a byte is two nibbles, a high order nibble and low order nibble.

8.2. Nibble Deletion

At first thought you might think that deletion would be an exact reverse of nibble insertion. However, the problem with that is that it would make moving data a nibble at a time impossible. There may be very valid reasons why might want to shuffle all the nibbles to left by an odd number of nibbles. So that is what MQEdit will do, regardless of whether the cursor is at the first or second nibble the data is shuffled to the left and, if necessary, padded with a 0 at the end of the string.

So, for example hitting delete at this point

```
12345678
  ^
```

would yield this:

```
12456780                            * MQEdit does this
  ^
```

hitting delete again would yield this:

```
125678                              * MQEdit does this
  ^
```

Of course if the hexadecimal string is part of a fixed length field, for example a Message Id or Correlation Id, the field does not reduce in length, the trailing zeros remain.

Chapter 9. Copy and Paste

MQEdit provides a number of powerful copy and paste features, from the simple copy and paste of characters to the interpretation of hexadecimal bytes.

9.1.1. Select All

Pressing this action will select the entire displayed text. Essentially the selection start will be at the start of the message and current cursor position will be placed at the end of the text. Note that if it is your intention to select the entire message then you should first make sure that MQEdit has retrieved the entire message. The 'select all' action makes it easier to copy or delete the entire message text. This action can be issued directly from the keyboard by pressing Ctrl-a.

9.1.2. Copy

Suppose the user wants to copy a Message Id and selects the field.

```
Message Id   :414D51204E5450474331202020202020898DE55420007202
```

When the user presses 'Copy' what do they want copied to the clipboard buffer? Do they want the characters that are displayed or the bytes they represent? Of course it is not easy for MQEdit to know until the data is pasted, then the context may give some clue.

So, to account for this MQEdit employs two clipboards, one containing text characters and one data containing bytes. The data clipboard is only used internally when copying/pasting between MQEdit windows all interactions with other applications can only use the text clipboard buffer.

To demonstrate these two areas try copying the text above in MQEdit. You would see a message at the bottom of the screen describing what was copied into the two areas.

```
Copied to Clipboard  : Text(48 bytes) Data(24 bytes)
```

Perhaps not surprisingly, since this is hex data, the amount of hex characters is exactly twice the number of hex bytes. However, this ratio will not always hold. For example, suppose you had copied the entire line you would get the following message:

```
Copied to Clipboard  : Text(62 bytes) Data(24 bytes)
```

You can see that the number of characters has clearly increased but the amount of message or message descriptor data has stayed the same.

So, we have our two clipboard buffers. but how do we control what gets pasted?

9.1.3. Paste

As we explained above the text buffer is always used when pasting in to external applications since this is the only clipboard that any other application knows about. However, if you are pasting in to an MQEdit window then MQEdit tries to make an intelligent choice about which to use. There is also a decision to be made about how to interpret the data.

Suppose the clipboard contained the data:

```
Copied Data
```

Then it is clear that this is just character data and should be treated as such.

However, suppose the buffer contains the characters:

```
54657374206D657373616765
```

Here it is clear that this is hex data. So, the buffer could represent something else.

This could be interpreted as the string:

```
Test message
```

since that is what those hexadecimal bytes represent on a ASCII system. In fact MQEdit allows you to organise the hex data however you like.

For example, the following string is the same bytes as the earlier example just in a different format:

```
54657374 206D6573 73616765
```

This can be useful when copying data from a dump file, or perhaps a trace file or a QLOAD output file.

So, MQEdit chooses which clipboard buffer to use and how to interpret the data based on the field you are pasting in to as follows:

Field Type	Paste Operation
Character Data	Uses DATA buffer if available otherwise use TEXT buffer If the paste buffer appears to be hex characters then MQEdit will ask the user whether they would like the data converted to the equivalent bytes
Hex Fields	Uses DATA buffer if available otherwise use TEXT buffer
Any other field type	Uses TEXT buffer only

If you want to force the editor to always paste the text buffer without any interpretation then you can hold down the 'Alt' button while doing the paste and the editor will do just that.

What we have discussed so far is pasting data in to a field. You will perhaps not be surprised to hear that the paste operation is limited to the size and data type of the field. For example, you can not paste the characters 'ABC' into a numeric field.. As far as length is concerned the editor will try to paste as much as it can. So, if the field is fixed length then the data will be pasted until the end. If the field is growable then the field will grow as required.

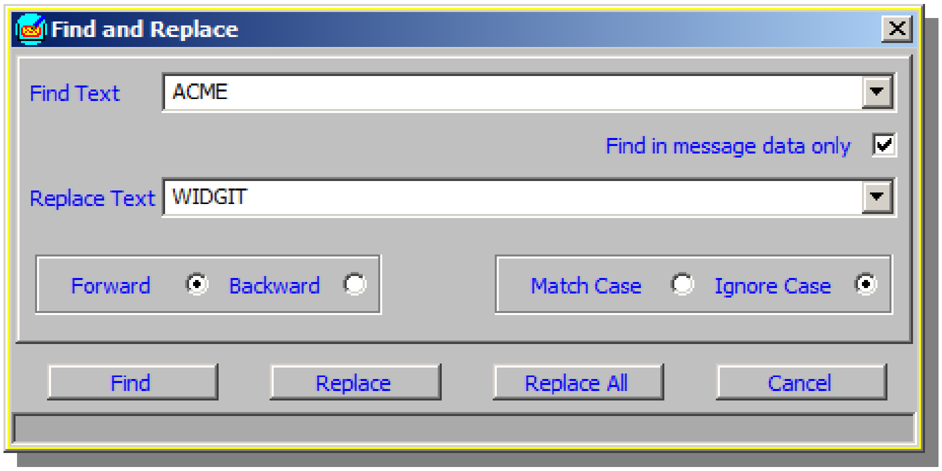
However, suppose you want to paste data that spans a single field. For example, suppose you want to copy an entire MQMD from one window and paste it over the top of another. Well, that's where 'Paste All' is useful.

9.1.4. Paste All

Unlike the normal Paste action, Paste All is not limited to the boundaries of a single field. Paste All requests that the paste buffer is treated as bytes and is just pasted at the target point with no regards for fields, field types or their boundaries. It is therefore very useful for copying whole blocks of data. However, be aware that it is very easy corrupt large portions of your message if you get the target point wrong. Of course, luckily you have the Undo button!

Chapter 10. Find and Replace Dialog

The 'Find and Replace' dialog operates much like the find/replace dialogs you find in a myriad of other editors so you shouldn't have any trouble driving it. We shall just point out the less than obvious features.



Find Text: ACME

Find in message data only ☑

Replace Text: WIDGIT

Forward ● Backward ○

Match Case ○ Ignore Case ●

10.1. Find Text

This text must be a standard ASCII string. However, when searching through a message the text will be converted to whatever codepage that element of the message is in. This means that you can easily find strings in an EBCDIC message.

One limitation is that all characters you are searching for must exist in both the Windows codepage and the target message.

The find operation will search the text as it is displayed. This means that if it is displayed in hex then it will search the hex, if you are searching in formatted view then it will search that. Equally it will search at the detail level you have selected.

The last few find text strings entered will be remembered across restarts of MQEdit.

10.2. Replace Text

In a similar fashion to the Find Text field the Replace Text field will be converted as required. So, you can Find and Replace text in an EBCDIC message with ease.

The last few find text strings entered will be remembered across restarts of MQEdit.

10.3. Find in message data only

The find dialog will find text anywhere in the output. This can be useful if you just want to find some text. However, since the find will also include non-editable fields this option is provided to allow you to restrict the search to just the message data.

When using 'Replace' or 'Replace All' the editor will only replace text in the data area anyway so this option is ignored.

10.4. Replace

In order to use Replace the cursor must be at text which matches the Find text. The replace button will replace the text under the cursor and, if successful, find the next occurrence of the Find Text.

The replace text can be empty which effectively allows you to delete the found text.

The replace might not be successful if:-

- The cursor is not at the Find Text
- The text is not an editable field
- You are trying to replace a field with an invalid character. For example, trying to put a letter in a number field.
- The Replace Text is the same as the Find Text!
- The field can only be changed in Expert Mode and that is not currently switched on.

The last few replace text strings entered will be remembered across restarts of MQEdit.

10.5. Replace All

Replace All will ripple through the entire message and replace all occurrences of the text with the replace text provided that none of the replaces fails for the reasons given above.

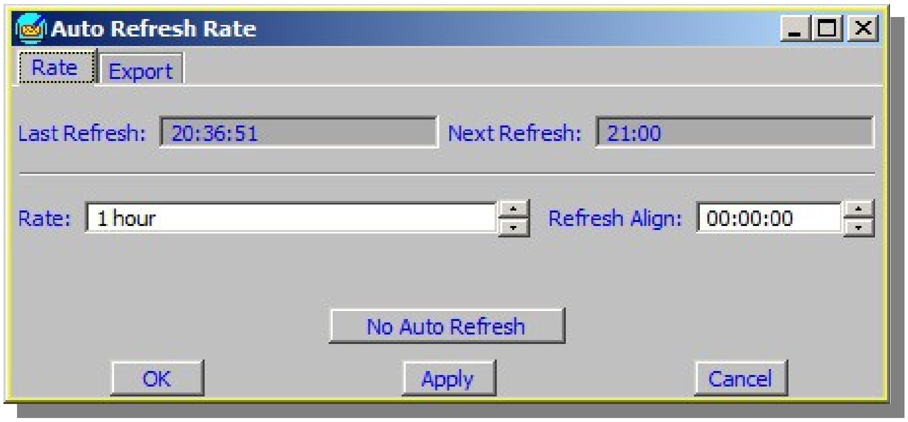
If the message is large this can be a lengthy operation. During the 'Replace All' MQEdit will display the current progress in a progress bar on the bottom right of the edit pane. The window which normally shows the cursor offset or line number will instead show a progress bar. The progress bar will remain after the 'Replace All' operation until you move the cursor.

The window will show how many occurrences were changed.

Chapter 11. Auto Refresh Dialog

This dialog allows you to set an auto-refresh time for the message list. This can be useful if you want to keep an eye on the message activity on the queue. Bear in mind though that it is just a snap-shot. If the queue is active then there is no guarantee that you would see all messages nor that if you were to select a message that it would still be on the queue.

The dialog is displayed when you select *'Options-Auto Refresh'* from the queue list or Queue Manager dialog. You can set a different refresh time interval per dialog type.



Last Refresh: 20:36:51
Next Refresh: 21:00
Rate: 1 hour
Refresh Align: 00:00:00

The dialog allows you to specify a refresh rate and, if required, a refresh align time. The rate is how often you would like the list refreshed. In this example we are saying every hour. The refresh align time field allows you to specify an alignment time. By default it has the value of 'None' indicating that there is no alignment. However, you can specify a time as in the example above. Please see "Time Interval" on page 137 for a description of the time format you can enter.

The align time is one of the times you would like a refresh to occur on. So, in this example we have specified a time which is 'on the hour' so this dialog will refresh every hour on the hour. We could have specified, for example, that we wanted to refresh daily at 9am. Aligning the refresh intervals to specific times of the day is most useful when combined with auto refresh exporting (which is explained in the next section). By aligning the export times you can more easily compare results of successive days and look for differences or trends.

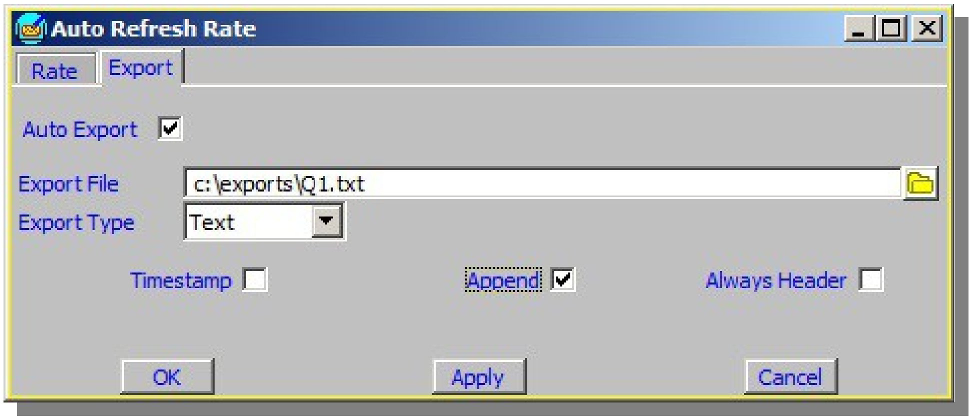
You may only use refresh alignment if the refresh rate is exactly divisible into 24 hours. If you specify a rate which is not divisible then the refresh align field will be disabled and the value of 'Unavailable' will be shown.

It is possible to set a very small refresh interval, for example 1 second, which is fine for particular tests on a development machine but probably not a good idea on a production system. Remember that refreshing a dialog involves message exchanges with the Queue Manager and is therefore not an insignificant cost particularly when there are a lot of messages involved in the request.

The 'No Auto Refresh' button is a quick and simple way of switching off auto refresh rather than having to type in a zero rate.

11.1. Auto Refresh Export Fields

The Auto Refresh dialog also gives you the ability to have an export file generated each time the data is refreshed. This is a rather specialised use of export but can be useful when you wish to capture a series of definitions. The resultant file can then be post-processed by another program to look at the trends. When exporting you can choose from three file formats, for post processing the formats of CSV or XML are likely to be the most useful.



Auto Export ☑

Export File c:\exports\Q1.txt

Export Type Text

Timestamp ☐ Append ☑ Always Header ☐

The fields related to auto refresh export and their meanings are as follows

11.1.1. Auto Export

Controls whether the dialog will write to the auto export file or not.

11.1.2. Export File

The export file format can contain character inserts which are described in "File Name Inserts" on page 136 . The button immediately to the right of this field will show a file selection dialog.

11.1.3. Export Type

Determines what type of export file will be generated.

11.1.4. Timestamp

Controls whether date and time columns are added to the exported data.

11.1.5. Append

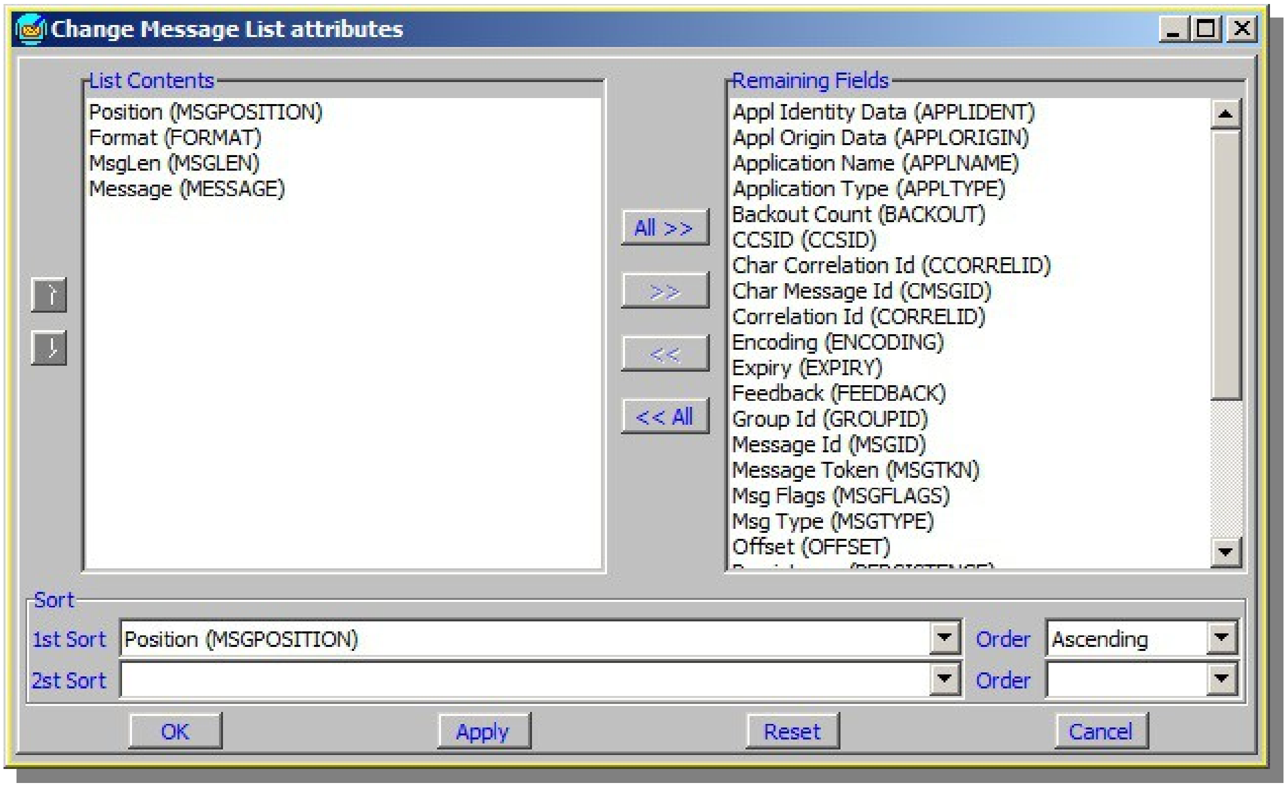
When selected any data will be appended to the end of the file if it exists. If not selected then any existing file will be over-written.

11.1.6. Always Header

Controls whether the column headings are always exported or not. If not set then the header is only written to the first line in the file.

Chapter 12. Alter list dialog

This dialog allows the user to change the list of attributes displayed in a list window. It is invoked either by selecting *'Alter list'* from the pop-up menu on the message list window.



List Contents
- Position (MSGPOSITION)
- Format (FORMAT)
- MsgLen (MSGLEN)
- Message (MESSAGE)

Remaining Fields
- Appl Identity Data (APPLIDENT)
- Appl Origin Data (APPLORIGIN)
- Application Name (APPLNAME)
- Application Type (APPLTYPE)
- Backout Count (BACKOUT)
- CCSID (CCSID)
- Char Correlation Id (CCORRELID)
- Char Message Id (CMSGID)
- Correlation Id (CORRELID)
- Encoding (ENCODING)
- Expiry (EXPIRY)
- Feedback (FEEDBACK)
- Group Id (GROUPID)
- Message Id (MSGID)
- Message Token (MSGTKN)
- Msg Flags (MSGFLAGS)
- Msg Type (MSGTYPE)
- Offset (OFFSET)

Sort

1st Sort: Position (MSGPOSITION) Order: Ascending

2st Sort: Order:

Figure 3: Alter List Dialog

The dialog displayed shows two sets of the attributes, the **List Contents** in the left hand column and the **Remaining fields** in the right hand column. Attributes are moved either by selecting them and pressing the arrow buttons or double clicking on items. To move all the attributes use the *'All >>'* and *'<< All'* buttons.

The attributes in these lists have a text description followed by the field name in brackets. This field name can be used in filter expressions (see "Filtering" on page 54 for more information).

When a field is added to the List Contents it will be placed before the first selected item if any. If none of the fields are selected then it will be put at the end of the list. At any time fields in the List Contents can be reordered by selecting a single entry and pressing the Up or Down arrows appropriately.

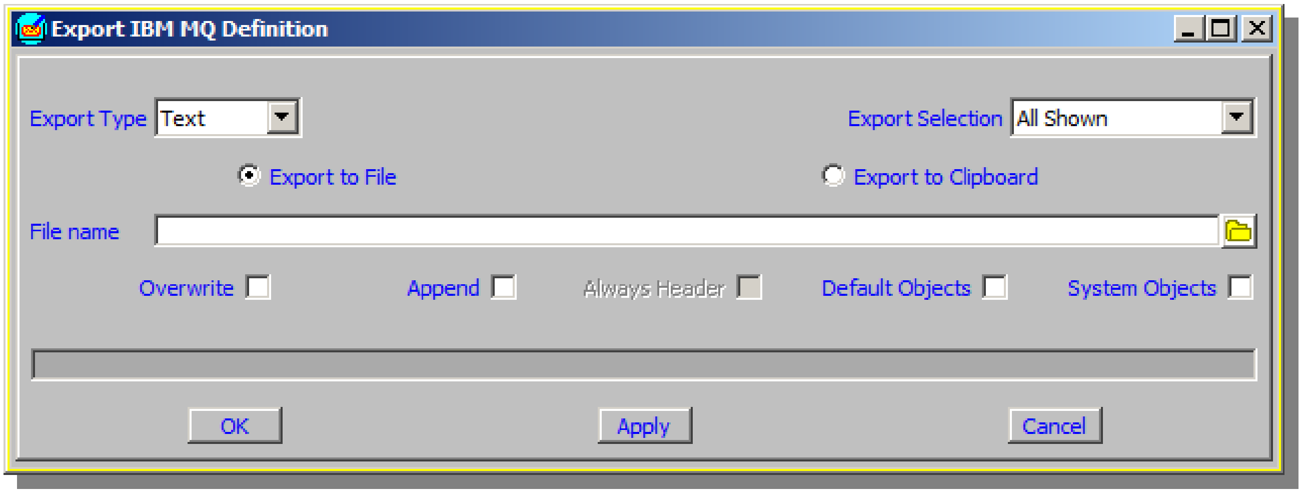
The default sort fields can also be specified using this dialog. Select the fields required for the major and minor sort fields and the direction of the sort. A list must have a major sort field but it does not have to have a specified minor sort field.

Pressing the *'OK'* or *'Apply'* buttons will cause the new attributes to be adopted by the list. By default the first attribute will be used as the sort field for the list. However, if an item is selected in the List Contents column when the *'OK'* or *'Apply'* button is pressed then that field will be used as the sort field. This simple technique is overridden if the user uses SORT functions in the filter window or clicks on the list titles.

As the number of columns is increased you may find that the complete field title is not displayed in order to save space on the dialog. By hovering the mouse over the column title a tooltip will be displayed which will give the complete name of the field and the field name which can be used in filter expressions.

Chapter 13. Export Dialog

From various places you can invoke the Export Dialog. This can be very useful for a number of reasons, and, as such, the export can take a number of formats.



Export Type: Text

Export Selection: All Shown

● Export to File ○ Export to Clipboard

File name

Overwrite ☐ Append ☐ Always Header ☐ Default Objects ☐ System Objects ☐

13.1. Fields

The export dialog presents a number of fields, these are:

13.1.1. Export Type

This field controls the output format of the export. The choices are:

Export Type	Meaning
Text	Export the fields in text format. Suitable for importing into other documents
CSV	Comma Separated Value useful for importing into an application such as a spreadsheet.
XML	XML useful for a variety of post processing.
MQSC	MQSC format it only available for some dialogs. It will generate an MQSC command which represents the objects values.

13.1.2. Export Selection

This field controls what is exported. The choices are:

Export Selection	Meaning
All Shown	Export all the objects
Selected	Export only those objects that are selected

13.1.3. Export to File or Export to Clipboard

A choice is presented as to whether the data should be exported to the file name given below or directly to a clipboard for pasting into another application.

13.1.4. File Name

The name of the file the program will write to. The file name can contain character inserts which are describe in “File Name Inserts“ on page 136.

13.1.5. Overwrite

When checked the program will overwrite any previous version of the file. Without this option checked the program will prompt the user for an overwrite decision if the file already exists.

13.1.6. Append

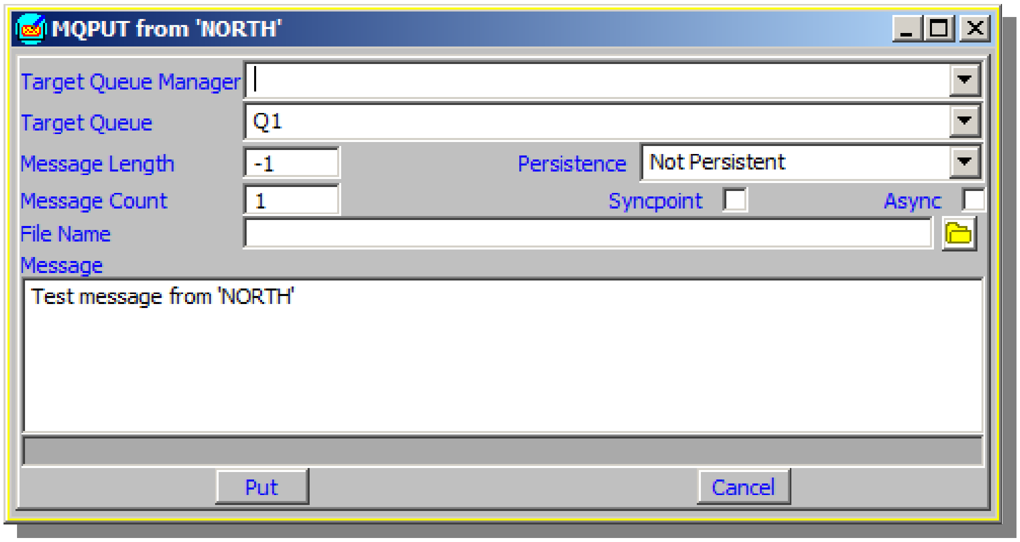
By default the program will overwrite any existing file. However, with the append box checked the data is written to end of any existing information in the file.

13.1.7. Always Header

This option is used to specify whether a header should always be written when appending to a file. If unchecked the header will only be written at the top of the file.

Chapter 14. Put Message Dialog

This dialog allows the user to put one or more simple text messages to a queue. It can be useful to generate some test messages and give a rough idea of performance.



MQPUT from 'NORTH'

Target Queue Manager
Target Queue: Q1
Message Length: -1
Persistence: Not Persistent
Message Count: 1
Syncpoint ☐
Async ☐
File Name
Message: Test message from 'NORTH'

14.1. Fields

The fields available are:

14.1.1. Target Queue Manager

The name of the target Queue Manager.

14.1.2. Target Queue

The name of the target Queue.

14.1.3. Message Length

The length of the required message. If this value is -1 then the length is given by the File Name or Message field.

14.1.4. Persistence

Whether the message should be persistent or not.

14.1.5. Message Count

How many messages should be sent.

14.1.6. Syncpoint

Whether the message should be put in or out of syncpoint. This option, when used in conjunction with 'Message Count' allows the user to get a feel for the effect of putting persistent messages under syncpoint.

14.1.7. Async

Whether the Async put should be used. Async put is an option available on the MQPUT call over a client link which improves put performance.

14.1.8. File Name

If specified then the entire contents of the file will be sent as a single message.

14.1.9. Message

A simple text message can be typed here.

14.2. Actions

14.2.1. Put

Each time the 'Put' button is pressed MQEdit will attempt to put the selected number of messages to the target queue.

If the puts fail then the status window will contain a message explaining the failure. If the puts works then a message will be displayed giving an idea of how fast the MQPUT calls were. The more messages you send the more accurate the timer will be. This simple measurement can provide a useful comparison of the relative speeds of the various options such as persistent or non-persistent messages, in or out of syncpoint for example. It can also give you a feeling for the speed of the network when running over a client connection.

14.2.2. Cancel

This button will close the dialog.

Chapter 15. Filtering

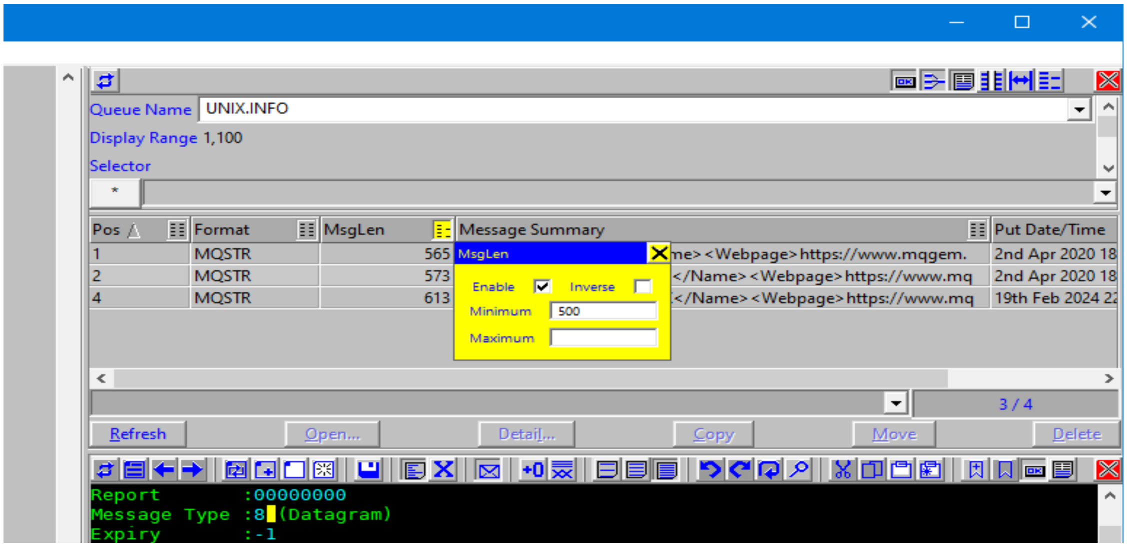
Sometimes you may want to filter the message list so that only certain types of messages are shown or highlighted. There are a number of ways you can reduce the messages shown. For example by Message Range or by Search Text. However, you can also use filters which allow you to change the display based on the attributes of the message. This can be useful when you can't see the wood from the trees. For example, you might want see message put only by a particular user or particular application. Or perhaps just look at the persistent messages. One key thing to remember is that all filtering is done in MQEdit itself, not at the Queue Manager. So, if you want to filter the entire queue you need to ensure that the message range is large enough to include the messages you are looking for. You can also do some filtering at the Queue Manager by using the Selector String available in the fields at the top of the message list..

MQEdit provides two ways of doing filtering, known as Column Filters and Filter expressions. There are advantages in each approach and which you use is entirely up to you. The advantages of each are:

- **Column Filters**
  Very simple click&display interface
  Show counts of enumeration fields. For example, you can easily see how many messages have a priority less than 5
- **Filter expressions**
  Very flexible, can be use to modify the display such as changing colours, adding and removing columns
  History of previous filters stored for later selection

You can, of course, use both mechanisms at the same time. If you do then the filter expression will be applied first, followed by any column filters. The numbers at the bottom-right hand of the dialog will show the results after both types of filtering have been applied.

15.1. Column Filters



Queue Name UNIX.INFO
Display Range 1,100
Selector

Pos	Format	MsgLen	Message Summary	Put Date/Time
1	MQSTR	565	me> <Webpage> https://www.mqgem.	2nd Apr 2020 18
2	MQSTR	573	</Name> <Webpage> https://www.mq	2nd Apr 2020 18
4	MQSTR	613	</Name> <Webpage> https://www.mq	19th Feb 2024 2

MsgLen
Enable ☑ Inverse ☐
Minimum 500
Maximum

3 / 4

```
Report       :00000000
Message Type :8 (Datagram)
Expiry       :-1
```

On the right-hand side of each column heading is a button which signifies that this column can be filtered. The icon on the button can be displayed in two ways. If both sides are the same then the filter is inactive. If there are less lines on the right-hand of the icon then the filter is active. In this case we have clicked on the 'MsgLen' filter and requested messages that are at least 500 bytes in size from the list. This will automatically enable the filter. All filtering is done locally, MQEdit will not go back to the Command Server for new data. Press Refresh on the list if you want a new set of data.

You can have as many column filters active as you like although only one filter window can be visible at any time. Selecting another column filter will close any previous column filter window for this dialog.

You can set the colour of the filter window by selecting *'View-Set Colours...'* from the main window. The filter windows colours are set by changing the 'Menu background' and 'Menu Foreground' colours. For information about setting colours please see Chapter Chapter 27.Colour Dialog on page 134. You can choose not to display column filters on a dialog if you wish by selecting *Options-Hide Column Filters* from the context menu.

15.1.1. Column Filter Types

As we know the different columns show different types of data. Each type of data will show a slightly different column filter window although the principals are the same. The following types are supported:

- **Character String such as Reply Q, Application Name etc**
  You can type in a single string of characters. You can choose whether MQEdit should merely search for these characters within the column or whether it should match the characters the with column entry. So, for example, suppose we entered the string **order** in the Application Name column filter. It would display messages put by applications with names like "**Invoice Order App**" or "**Web Order App**" if we had selected contains. But it would not display either of those if we said matches. There is also the option to select 'Match Case' to be even more specific.

  We can go further with a *match* type filter and add wildcards. So, for example, we could enter the characters ***Order App** and this would show all the messages put by applications that ended with "**Order App**". We can also use **?** indicating a single character. So, for example, **????**, would match with any four character application name.
- **Hex fields such as Message Id, Correlation Id**
  Hex strings are very similar to normal string except that the concept of 'Match Case' does not apply. You must enter a hex string and possibly wildcards for the match.
- **Numbers such as Backout Count, or CCSID**
  To filter by a number you are given a minimum and maximum value and for most numbers a list of the used values. If you enter a number in the minimum field then only entries with a value greater than or equal to this value will be shown. Similarly a number entered in the maximum field will cause on entries with a value less than or equal to this number to be shown.
  Most numeric fields will also display a list of the used values and how many of each are in use. Whether this is available depends on the type of field, whether it is status or whether it is definitional. For example, it is not available on 'Message Length'. If it is applicable though this feature allows you to easily select the values that interest you. Alternatively selecting 'Any value' will remove any selection based on a particular value and show entries of any value. By default only the 'in use' values are shown in the list of values but selecting 'Show All' will show any values that MQEdit has seen but that are not in the currently displayed list. Pressing 'Refresh Values' will cause MQEdit to forget any value that is not currently in use by this list.
- **Enumerations, such as Message Type**
  There are essentially two types of enumeration fields in MQ. Those that have a small number of choices, such as message type and those that have a large number of choices, such as feedback code.
  - **Small numbers of choices (15 or less)**
    A simple list of checkboxes is displayed. Each checkbox is followed by a number which is the count of items in that list. All possible choices are shown regardless of whether the list itself actually contains entries of that type.
  - **Large numbers of choices**
    If there are a large number of possible choices for this field then MQEdit will, by default, only show the choices that currently exist in the displayed list. You can select the 'Show All' checkbox if you really want to see all possible values. The values are shown in a scrollable list.
- **Combined Date&Time fields**
  Traditional MQ Date and Time fields are just a set of characters and are treated as such. This can be useful if you want to do wildcard processing etc. However, since the combined Date&Time fields are an actual time their column filters can treat them as such. For more explanation please refer to 'Date&Time Column Filters' on page 56.
- **Flags, such as Report Options**
  Flag fields are very similar to normal enumeration fields except that the field counts are handled differently. A single list entry could have more than one flag active. It follows therefore that the sum total of the enumeration counts will not sum to the number of items in the list.

The message list display will be updated immediately when most of the filter fields are changed. For example when an enumeration choice is selected or deselected. However, the application of the data entry fields, such as the search field and the minimum and maximum values is not immediate. For these fields MQEdit will only update the list when no data entry has occurred for approximately two seconds. Alternatively you can signal that editing has completed by pressing *<ENTER>*. In which case the message list will be re-filtered immediately.

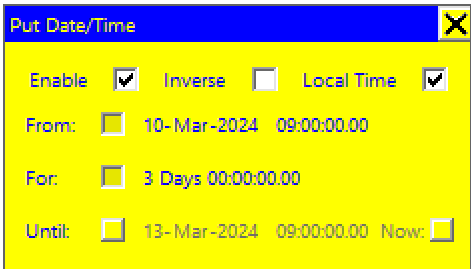
As mentioned before you can have as many Column Filters enabled as you wish. You can select '*Options-Remove Column Filters'* from the context menu (or message list toolbar) to quickly remove all column filter filtering. MQEdit will still remember the last filter values used it is just that the column filter itself will not be enabled.

15.1.2. Column Filtering Positioning

MQEdit will attempt to put the filter window in a suitable place next to the column being filtered. If you scroll the message list or resize it then the filter window will move accordingly. If you wish to have the filter in a particular place though then you can click on the titlebar of the window and move the filter window to your preferred location. MQEdit will remember the relative position of the window and the filter window will continue to be moved in accordance with size and position changes to the dialog.

15.1.3. Date&Time Column Filters

Because a Date&Time field represents an actual time we can filter by physical timestamps. Consider the following:



Put Date/Time

Enable ☑ Inverse ☐ Local Time ☑

From: ☑ 10-Mar-2024 09:00:00.00

For: ☑ 3 Days 00:00:00.00

Until: ☐ 13-Mar-2024 09:00:00.00 Now: ☐

This filter will show only those items in the list which were put on or after 9am March 10th 2024 but before 9am March 13th 2024. Note that the 'From' date is inclusive, but the 'Until' date is not.

The Date&Time Column Filter can be used in a number of different modes. The most obvious difference depending on whether the 'From', 'For' or 'Until' fields are selected. These modes are described in the following table:

From Selected	For Selected	Until Selected	Effect
X			Shows items from the given date until 'now'
X	X		Shows items from the given date for an elapsed time period.
	X		Has no practical meaning. All list items will be shown
		X	Shows items until the given date.
	X	X	Shows items for a time period leading to an explicit date.
X		X	Shows items from the given date up to, but not including, the Until date.

So, depending on which fields are 'active' the dialog can be used to select items on the list conforming to a variety of selections.

The fields themselves can be changed in a variety of ways. However they can only be changed while they are selected. Clicking on an individual element will place focus bars above and below the element. This allows you to know which field will be changed. You can change which element has focus by either clicking on a different element with the mouse or using the left and right keys. The value itself can be changed in any of the following ways:

- **Using the up and down keyboard keys**
  Holding down a key will continuously change the value
- **Using the mouse wheel**
- **Clicking on the focus bars.**
  Holding down the mouse will continuously change the value.

Note that you cannot have all three fields selected. However, if you have two fields selected when you are changing a value then the 3rd value will be changed accordingly. If an invalid value is entered, for example 30th Feb, then the field will be shown in red and the field will not be active in the selection.

15.1.3.1. Now

The Until time has a 'Now' button following it's value. This allows you to quickly revert the value of Until to the current time. In fact, while ever the Now button is pressed the 'Until' time will constantly be updated with the current time. This allows you to have an ongoing filter which shows you, for example, just what happened in the last 4 hours. Note that column filters only filter what is currently in the list, they do not cause data to be refreshed from the Queue Manager. Therefore, in order to do this, you must use it in conjunction with Auto-Refresh on the message list. Auto Refresh allows you to specify the interval you wish to refresh, for example, every 5 minutes. Choose a sensible refresh interval, unless you are testing, do not refresh your message list every second!

15.1.3.2. Local Time

You can choose whether the entered times should be treated as 'local' times or a time from the Queue Managers reference point which for put date/time is GMT. For example, something which happened at 9am my time or something which happened at 9am GMT.

15.2. Filter Expressions

At its heart a filter expression is merely a boolean expression which returns TRUE or FALSE for each message in the list which determines whether the message is displayed or not. However, filters can also be used to change the way the message is displayed.

Let's take a very simple example. Suppose we want to see just the non-persistent messages and highlight any message which is larger than 1000 bytes. We can use the following simple filter.

```
persistence = no & bg(msglen > 1000,yellow)
```

Those of you with a mathematical leaning will understand that filter immediately but for others it may take a little playing with. All you need to remember that at it's heart a filter is merely a boolean expression which evaluates to TRUE or FALSE for each message in the list. So, this filter is saying the expression is TRUE if 'persistent is 'no' AND the result of function *bg()* is TRUE. Well, *bg()* always returns TRUE as we'll see later. So, this expression will be TRUE for any message that is non-persistent. Which is exactly what we wanted. However, the calling of the *bg()* function clearly has another effect. b*g()* takes two parameters, the first is another boolean expression and the second is a colour, in this case the constant 'yellow'. I'm sure by now you can guess just how the *bg()* function works and could quite easily write your own filter which showed high priority messages in red or messages soon to expire in green. Filters are very flexible and powerful as we will see in a minute.

Where possible, filtering is performed against the data cached in the editor. In other words, typing in an expression and pressing the '*' filter button will not necessarily cause MQEdit to re-browse the messages. If all the information needed to satisfy the expression is already stored locally the filter will be performed on that information. Consequently the display of data is much faster. However, as a consequence the data displayed will not necessarily be the latest up-to-date data and users should periodically hit the *'Refresh'* button to download the latest information from the Queue Manager. If the filter expression contains attributes that are not cached then a request for more information will be sent to the Queue Manager.

At the bottom right hand corner of the screen are two numbers. The first number gives the number of messages displayed in the list, and the second number gives the number of messages returned from the queue manager. If filtering is not active, these two numbers will be the same, however, with filtering there may be fewer messages in the list than were returned from the queue manager. Bear in mind though that the right-hand number does not necessarily indicate how many messages are on the queue. This very much depends on the range of messages you have requested.

The expression rules are as follows:-

- **The expression is free-format**
  e.g. "2+4" and "3 + 4" are both acceptable
- **Constants**
  - Colours (used in the fg() and bg() functions)
    white, black, blue, red, pink, green, cyan, yellow, brown, gray, dblue, dred, dpink, dgreen, dgray
    The constants starting with 'd' are darker versions of the colour.
  - Booleans
    true, false
  - Console priorities (used in the csl() function)
    info, low, medium, high
- **Numbers**
  Numbers can be entered as:
  - Integer 12,1234
  - Real 1.3,12345.5
  - Hex 0xFAB,0x10
- **Strings**
  Strings can be any sequence of characters between ‘ or “ characters. Special characters can be included in the string using the ‘\<value>’ convention.
  - \\ Backslash. This should be used, for example, when specifying file names with back slashes in them.
    For example use **c:\\temp\\file.dat** not **c:\temp\file.dat**
  - **\n** New line
  - **\”** Double quote
  - \’ Single quote
  - **\a** Alert
  - **\b** Back space
  - **\f** Form feed
  - **\r** Carriage return
  - **\t** Tab
  - **\v** Vertical tab

  For example :-

  - "SYS.*"
  - 'ABC'
  - "This string has a \" in it"

- **Statements**
  A filter can consist of multiple statements joined together using a ‘;’ character. The value of the statements is the last statement in the list. For example the filter **status(“Hello World”);0** has the value 0.

- **Functions**
  Various functions, such as the *bg()* function we've already seen, are provided which can be called to do variety of tasks. Refer to the description of “Functions“ on page 62 for more information.

15.3. List Variables

Any field name of the message being displayed may be used in the expression. You only need specify enough of the name to ensure uniqueness need be specified. To see the identifier that should be used look in the *'Alter List'* dialog where the complete list of attributes and their field names are given. Alternatively if it's a displayed field you can hover the mouse over the column title and a tooltip window will be displayed giving the full name of the field and the field name which should be used in filter expressions.

It is not necessary to only use fields from the currently displayed list.

It is often useful to be able to temporarily see a value in the list without going through the process of actually changing the list items using *'Alter List'* described earlier. In this case you can follow the variable name with a '#' (hash) character. This will cause that field to be displayed if it's not already in the list. The columns will be displayed after the defined columns for the list in the order they were written in the expression. As soon as a filter is entered without this field followed by a hash then the field will be removed from the display.

So, for example the following filter will add the priority field to a message list but still display all messages

```
priority#;1
```

The effect of adding the ';1' to the end of a filter is to ensure that it always returns TRUE. If we had just used a filter of

```
priority#
```

Then only messages with a non-zero value of priority would be displayed.

Some other examples of value field names in the list are as follows:

Variable	Meaning
ApplName	Putting application
MsgType	Message Type
ReplyQ	Message Reply Queue
Format	Message Format
Expiry	Message Expiry time
Priority	Message Priority
Pri	Message Priority
Pr	Message Priority
PutTime	Time the message was put

15.4. Operators & Flow control

Normal operator operation and precedence apply :-

Operator	**Meaning**
+, -, *, /	Addition, Subtraction, Multiplication, Division The ‘+’ operator is also used to concatenate strings.
mod	Modulus
=, >, >=, <, <=, <>, !=	Boolean comparisons
&, |, !	Boolean AND, OR and NOT
&&, ||	Bitwise AND and OR[13]
==	String wild card matching The second operand is specified as a string containing the following wild card characters: - '*' Matches any number of characters (including 0) '?' Matches exactly 1 character.

- **Conditional branches**
  It is possible to take conditional branches using the following construct:-
  if (expression) {statements} [else {statements}]

- **Coercion**
  If two different operand types are involved in a sub expression the type of one or more of the operands will be changed. Most notably if strings are used in expressions other than comparisons then the string length is what is used.

  For example, the filter **"replyq"** is TRUE only if the message has a non-blank reply-to-queue. So for example the filters **replyq == "???"** and **replyq = 3** will both display the list of messages with only three character reply-to-queues.

  As another example, **SORT(replyq)** will sort the list in alphabetic reply queue name order but **SORT(replyq+1)**, since we're forcing the reply queue to be treated as a number, will display the list in order of the length of their name. A similar effect is achieved by using **SORT(+replyq)**.

- **Invoking other filters**
  If a filter has been given a name then it can be invoked from another filter using the expression **$<filter_name>**. Note that this means that filter names can not contain blank characters.

---

[13] Please note that the bitwise and logical operators are the other way round from languages such as ‘C’. This is due to the fact that most of the time the single operator, logical version, is required.

15.5. System Variables

System variables are predefined variables which are initialized to a value before being used in the filter.

- **_first** An integer which is set to 1 (or true) each time the filter is matched against a set of entries. This integer allows you to do some initial processing. Note that the filter should set the value of **_first** back to 0 once initial processing has completed with the statement.
  For example, **_first := 0;**
- **_last** An integer which is set to 1 (or true) only for the last entry in the list. The **_last** system variable is most useful for reporting results.
  For example, **if (_last) status("All done now");**
- **_time** An integer which contains the current time. This can be useful for calculating the different in time between successive invocations of the filter or for performing different processing in the filter depending on the day of the week or time of day.
- **_index** The index of the current item in the list starting at 0.
- **_items** The number of items in the list
- **_qmname** The name of the Queue Manager
- **_locname** The name for this location
- **_scan** The scan instance. This field can be used to discover how many times this current data has been scanned. This can be used in conjunction with **_rescan** to examine all data in scan 0 before deciding what to return in scan 1.
- **_rescan** Whether another scan should be performed. A filter can set this value to true to force a re-scan of the current entries. This can be useful to collate information about the data in the first scan, then during the re-scan the filter can use this collated data to make decisions about the data.

15.6. User Variables

User variables begin with the character '@'. They may be used to store information either between items in a list or between separate invocations of the list filter. Different data types of user variables are allowable by following the '@' by an optional character.

@x := 1	Declares x as an integer variable
@@x := 3.2	Declares x as a real or float variable
@$x := "Hello"	Declares x as a string variable String variables may only be a maximum of 200 characters.

As shown in the example above, assignments are made using the '**:=**' operator. Note the use of the colon character to distinguish the assignment from a comparison.

For example, **@x = 3** is comparing **x** with the value 3 whereas **@x := 3** is assigning **x** the value 3.

15.7. Functions

- **alarm(Exp)**
  Causes the application to alarm, always returns TRUE. If 'Exp' evaluates to TRUE then the application will issue an alarm BEEP every few seconds provided alarms are switched on for the application.

- **any(Exp1, Exp2, Exp3,........)**
  Takes 1 or more parameters. The **any** function is useful for displaying fields which can contain any value. For example a filter of **‘priority#’** will display a column for Message Priority but will only display those items for which the value of priority is non-zero, it is equivalent to ‘priority# <> 0’. There are times when you want to see the value displayed regardless of what it is. A filter of **‘any(priority#)’** will display those messages for any value of Priority.

- **beep(Exp)**
  Causes the application to beep, always returns TRUE. If 'Exp' evaluates to TRUE then the application will issue a BEEP.

- **bg(Exp, Colour)**
  Sets background colour, always returns TRUE
  If 'Exp' evaluates to TRUE then it sets the background colour of the object in the list
  The colour can be specified using the colour constant or can be an expression.

- **bgcell(Exp, Colour, Cellname)**
  Sets background colour of a single cell, always returns TRUE
  If ‘Exp’ evaluates to TRUE then it sets the background colour of just the specified cell in the list. For example, bgcell(msglen>1000, red,msglen) will set the background colour of just the Mesage Length for any messages larger than 1000 bytes.
  The colour can be specified using the colour constant or can be an expression.

- **date$([time [,format]])**
  Returns the date as a formatted string.
  This function takes 0, 1 or 2 parameters. If no parameters are given the current time is returned in the default format. If just the time parameter is passed then that time is returned in the default format. If both a time and format is given then the returned value is the time in a formatted string according to the following values for the format string.

Insert	Meaning
H	Two digit hour (24 hour clock)
HH	Hour (24 hour clock)
h	Two digit hour (12 hour clock)
hh	Hour (12 hour clock)
M	Two digit minutes
S	Two digit seconds
d	Two digit day of month
dd	Day of month including suffix eg.1st, 2nd, 3rd …
j	Julian day of year (zero based)
J	Julian day of year (one based)
m	Three character month name eg.Jan,Feb,Mar….
mm	Two digit month
mmm	Full month name eg. January, February,March…
P	AM/PM
p	am/pm
y	Four digit year
yy	Two digit year
D	Three character day of week eg.Mon,Tue,Wed…
DD	Full character day of week eg. Monday, Tuesday, Wednesday…
t	Simple time format eg. 18:14:03
\\**<char>**	Escape character sequence. eg. \\m will print 'm'

The default format is : **H:M d/mm/y** eg. 22:10 05/08/2016

- **day(time)**
  Returns day of the month represented by the time parameter. Returned value from 1->31.
- **day$(time, type)**
  Returns day, as a string, represented by the time parameter. The type parameter controls the format of the returned string.
  - **Type = 1**
    Sun,Mon,Tue,Wed,Thu,Fri,Sat
  - **Type = 2**
    Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday
- **fclose(fd)**
  Closes the file descriptor.
- **fg(Exp, Colour)**
  Sets foreground colour, always returns TRUE
  If 'Exp' evaluates to TRUE then it sets the foreground colour of the object in the list.
  The colour can be specified using the colour constant or can be an expression.

- **fgcell(Exp, Colour, Cellname)**
  Sets foreground colour of a single cell, always returns TRUE
  If 'Exp' evaluates to TRUE then it sets the foreground colour of just the specified cell in the list. For example, fgcell(msglen>1000, red, msglen) will set the foreground colour of just the Message Length for any messages larger than 1000 bytes.
  The colour can be specified using the colour constant or can be an expression.

- **findstr(String, SubString)**
  Searches for SubString in the String. If it is found the function returns a 1 based offset of the position. If the string is not found the function returns zero (FALSE). The search is case sensitive.

- **findstri(String, SubString)**
  Searches for SubString in the String. If it is found the function returns a 1 based offset of the position. If the string is not found the function returns zero (FALSE). The search is case insensitive.

- **flash(Exp)**
  If 'Exp' evaluates to TRUE then it will periodically flash the line, always returns TRUE

- **flashcell(Exp, Cellname)**
  If 'Exp' evaluates to TRUE then it will periodically flash specified cell, always returns TRUE

- **fopen(FileName [, filemode])**
  Returns file descriptor which can be passed into file routines, or 0 if open failed.
  If the file name contains the '\' character remember to use '\\' instead.

  Optional file mode parameter controls how the file is opened. Valid values are :-

"r"	Open for reading
"w"	Open for writing (Default if filemode not specified)
"a"	Open for append
"r+"	Open for reading and writing but file must exist
"w+"	Open for reading and writing. Any current file will be overwritten**.**
"a+"	Open for reading and appending
"b"	Open in binary mode
"t"	Open in text mode [Default if neither "b" or "t" specified]

- **fprint(File, Exp1, Exp2, Exp3,…….)**
  Writes the expressions to a file.
  The file parameter can either be a file name or a file descriptor returned from **fopen()**
  If the file name contains the '\' character remember to use '\\' instead.

- **fprintf(File, Fmt, Exp1, Exp2, Exp3,…….)**
  Writes the expressions to a file a formatted string.
  The file parameter can either be a file name or a file descriptor returned from **fopen()**
  If the file name contains the '\' character remember to use '\\' instead.

  For example the filter **fprintf("c:\\temp\\mqedit.out","%50s %d\n",format,priority)** on the message list dialog will write a file containing the format and priority of all local messages.

- **hidecol(ColumnName)**[14]
  This function will hide the names column from a list display. For example, **hidecol(qtype)** will suppress the display of the Queue Type column in a Queue List display.

- **hour(time)**
  Returns the hour portion of the time parameter

---

[14] This function is unusual in that it is not run against each entry in a returned list – instead it is run once, initially, when the filter is defined.

- **max(Exp1, Exp2)**
  Returns maximum value of the two expressions
- **min(Exp1, Exp2)**
  Returns minimum value of the two expressions
- **minute(time)**
  Returns the minute portion of the time parameter
- **month(time)**
  Returns month of the year represented by the time parameter. Returned value from 0->11.
- **mon$(monthindex, type)**
  Returns month of the year, as a string, represented by the month index (0->11) parameter. The type parameter controls the format of the returned string.
  - **Type = 1**
    Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec
  - **Type = 2**
    January, February, March, April, May, June, July, August, September, October, November, December
- **mqtime(Date, Time)**
  Accepts date in YYYY-MM-DD format and time in HH.MM.SS format and returns number of seconds since 1970 as a long.
- **msgtime(Date,Time)**
  Accepts date in YYYYMMDD format and time in HHMMSShh format in UTC time and returns number of seconds since 1970 as a long.
- **second(time)**
  Returns the second portion of the time parameter
- **showcol(ColumnName, Column)**
  This function will display the required column at the given column number. For example, **showcol(priority,4)** will display the PRIORITY values in the fourth column. If the Column value is zero or less then the column position is not changed.
- **sort(Exp)**
  Sorts the list in ascending order of the expression. Note that since this is an explicit sort it will effectively disable primary sorting by the user by clicking on the list titles.
- **sortd(Exp)**
  Sorts the list in descending order of the expression. Note that since this is an explicit sort it will effectively disable primary sorting by the user by clicking on the list titles.
- **sort2(Exp)**
  Specifies the secondary sorting expression in ascending order. Note that since this is an explicit sort it will effectively disable secondary sorting by the user by clicking on the list titles.
- **sortd2(Exp)**
  Specifies the secondary sorting expression ni descending order. Note that since this is an explicit sort it will effectively disable secondary sorting by the user by clicking on the list titles.
- **sound(Exp, Wave file)**
  Causes the application to play wave (.WAV) file, always returns TRUE. If 'Exp' evaluates to TRUE then the application will play the wave file.

- **status(Exp1, Exp2, Exp3,…….)**
  Writes the expressions to the status line of the dialog.
- **statusf(Fmt, Exp1, Exp2, Exp3,…….)**
  Writes the expressions to the status line in a formatted string.
- **str$(Exp1)**
  Return a string representation of whatever parameter is passed in.
- **substr$(String, Offset, Length)**
  This function returns the substring of the given String at the given offset and length. The start of the string is at offset 1.
  An Offset value of -1 requests that the function returns the substring from the end. For example substr$("ABCD",-1,2) will return "CD".
  A Length value of -1 requests that the function returns as much of the string as is left. For example substr$("ABCD",2,-1) will return "BCD".
  The substr$ function will return a maximum of 200 characters.
- **weekday(time)**
  Returns day of the week represented by the time parameter. Returned value from 0->6 where 0 represents Sunday.
- **yearday(time)**
  Returns day of the year represented by the time parameter. Returned value from 0->365.

15.8. Output Format String

Some functions, such as **fprintf** and **statusf**, allow data to be output in a formatted string. The format string is modelled closely on the ‘C’ language printf format string.

%[flags][width][.precision][type]

where :-

flags

-	Left aligned
+	Prefixes the output a + or – sign
0	Prefixes the output number with ‘0’
#	For **o**,**x**, or **X** fields with prefix with 0,0x or 0X respectively. For **g** or **G** fields it forces the output to contain a decimal point.

width

An optional positive integer specifying the minimum number of characters printed for this field.

precision

An optional positive integer specifying the number of decimal places, the number of significant places or the number of characters to be printed depending on the data type.

type

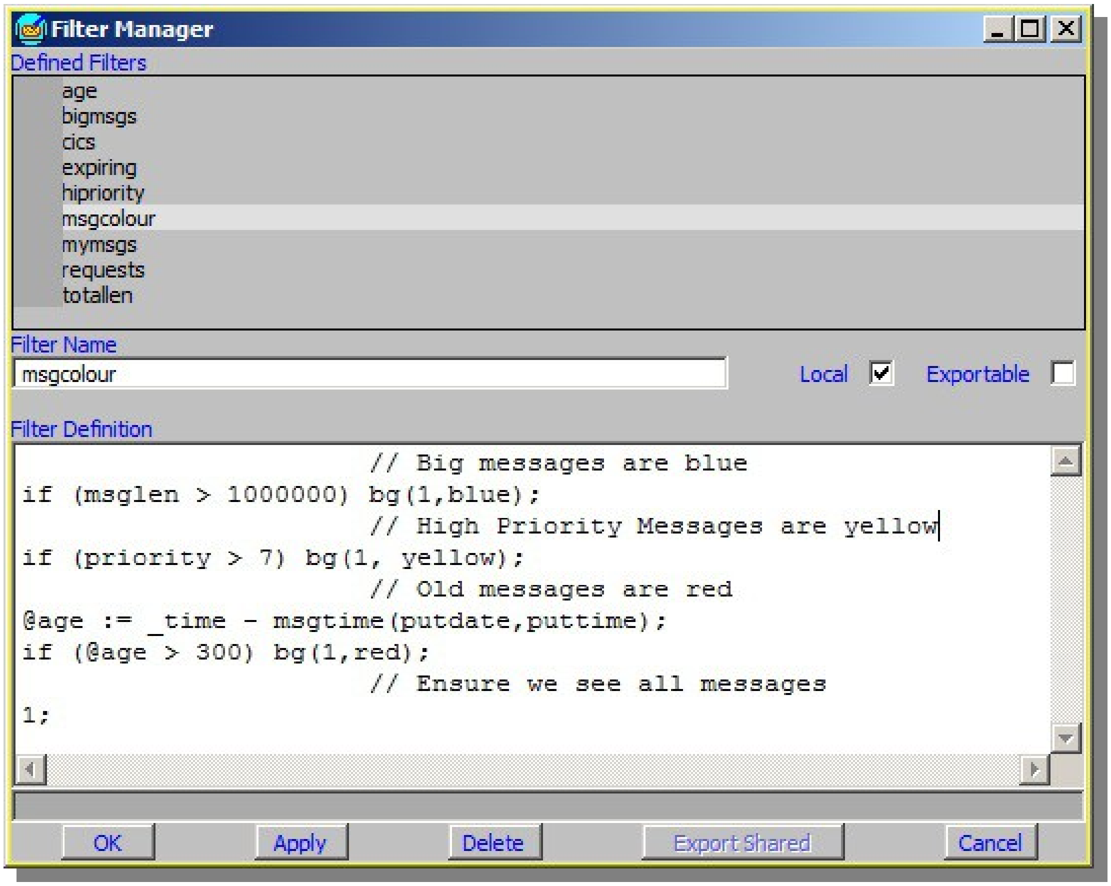
d,i	Signed integer
o	Octal integer
u	Unsigned integer
x	Hexadecimal integer using lowercase characters
X	Hexadecimal integer using uppercase characters
e	Real number in the format [-]d.dddd e [- | +]ddd
E	Real number in the format [-]d.dddd E [- | +]ddd
f	Real number in the format [-]dddd.dddd
g	Real number in either **e** or **f** format depending on which is smaller
G	Real number in either **E** or **f** format depending on which is smaller
S	String

Examples :-

Specification	Output
statusf(“%d” , 2)	2
statusf(“%05d” , 2)	00002
statusf(“%.7s”,”This is a test”)	This is
statusf(“%s = %d”,”Value”,7)	Value = 7
statusf(“%s = %d”,”Value”)	Value = %d *Note that there aren’t enough parameters to fill the inserts*

15.9. Filter Manager

There may be a number of filters which are either used frequently or that are fairly complicated. As you will see in the examples that follow some filters can be many lines long. For these filters it may be more convenient to define the filter globally and assign a name to it. The name should not contain blank characters. To access the filter manager use the *'Action-Filters...'* main menu item or by pressing **<Ctrl-m>**. This will display the following dialog.



Defined Filters

- age
- bigmsgs
- cics
- expiring
- hipriority
- msgcolour
- mymsgs
- requests
- totallen

Filter Name: msgcolour

Local ☑ Exportable ☐

Filter Definition

```
                      // Big messages are blue
if (msglen > 1000000) bg(1,blue);
                      // High Priority Messages are yellow
if (priority > 7) bg(1, yellow);
                      // Old messages are red
@age := _time - msgtime(putdate,puttime);
if (@age > 300) bg(1,red);
                      // Ensure we see all messages
1;
```

Figure 4: Filter manager Dialog

You can see that a number of filters have been defined and the definition of 'msgcolour' is shown. The idea of this filter is that it will set the background colour of the message depending on the attributes of the message. To use this filter a filter of '$msgcolour' should be specified in the message list window.

As well as a name and a definition a filter has two other attributes, Local and Exportable. A local definition is one which is defined locally to this instance of MQEdit, it's definition lives in the MQEDIT.CFG file. If a definition is not local then it is 'shared'. This means that it's definition was read in from the shared filter file. Please see “Shared Filters“ below for a description of shared filters. Only 'Local' filter definitions can be created or altered. The other attribute, Exportable, indicates whether this filter should be exported to the shared filter file when the 'Export Shared' button is pressed.

Pressing 'OK' or 'Apply' will add or update the currently displayed filter and cause any dialogs using this filter to update their display.

The Delete button will delete the filter currently selected in the list. Before deletion however the user is shown a confirmation dialog.

15.9.1. Comments

As you can see from the example above you can comment your filters by starting with the characters //. This marks the whole line as a comment which allows you to annotate your filter.

15.9.2. Shared Filters

Sometimes it can be useful to be able to generate a set of filters and have those filters used by other machines using MQEdit. These are known as shared filters. The concept is that you generally have one 'master' MQEdit which exports the shared filters and you have many 'reading' instances of MQEdit which just import the filters.

Whether an instance of MQEdit is a 'master' or not is controlled by the 'Import shared filters' option in the 'Lists' tab of the Preferences Dialog. If this option is checked then MQEdit will import filters it finds in the shared filter file which is also configured in Preferences Dialog.

If the 'Import shared filters' option is not checked then MQEdit can export certain shared filters to the shared filter file.

To generate shared filters

To create a set of shared filters you do the following actions.

1. Ensure that 'Import shared filters' in the 'Lists' tab of the Preferences Dialog is not checked
2. Fill in the name of the 'Shared Filter File' you wish to use in the 'Lists' tab of the Preferences Dialog.
3. Bring up the Filter Manager
4. Define or Alter your filters as required
   If you wish a filter to be shared then ensure the 'Exportable' checkbox is checked and press the 'Apply' button.
5. The list of filter names will show shared filters with an [S] in the left most column
6. Once the list of filters is correct press the 'Export Shared' button.
   This will generate the shared filter file and report how many filters were exported[15].

To use shared filters

To use a shared filter file you first need access to file containing the shared filters. This could be exactly the same file as was generated or the file could be copied locally. Similarly the file could be made available on a shared file server.

The steps you should take are:

1. Ensure that 'Import shared filters' is checked in the 'Lists' tab of the Preferences Dialog.
2. Fill in the name of the 'Shared Filter File' you wish to use in the 'Lists' tab of the Preferences Dialog.
3. Restart MQEdit
   Now, every time MQEdit starts the shared filters file will be read ensuring that MQEdit has the latest set of definitions.

---

[15] Ensure that the shared filter file value is correct since this action will overwrite the contents of the file.

15.10. Filter Examples

On the basis that the best way to see what effect filters have is the try them, here are a number of examples (some of use and others purely for demonstration purposes).

- **ReplyQ = "IMS"**
  Show only messages with a reply queue of "IMS"
- **ReplyQ == "IMS*"**
  Show only messages with a reply queue beginning "IMS"
- **ReplyQ == "*IMS"**
  Show only messages with a reply queue with a name ending "IMS"
- **ReplyQ == "*IMS*"**
  Show only messages with a reply queue with "IMS" somewhere in their name
- **ReplyQ == "????"**
  Show only messages with a reply queue with 4 characters names
- **ReplyQ = 4**
  Show only messages with a reply queue with 4 character names
- **ReplyQ < 8**
  Show only messages with a reply queue with less than 8 character names
- **ReplyQ**
  Show only messages which have a reply queue
- **ReplyQl#**
  Show only messages which have a reply queue and display the reply queue as a column
- **sort(msglen)**
  Sort the list in order of Message Length
- **sortd(applname)**
  Sort the entries based on the Application Name in descending order
- **beep(priority = 9)**
  Beep if there is a priority 9 message on the queue
- **@Age# := _time – msgtime(putdate,puttime);**
  Display a new column called Age which gives the age of the message in seconds.
- **Total number of bytes of all the messages**
  This example adds up the msglen value for each message. It then displays the total on the status bar after processing the last entry.

```
if (_first)
{
  @total := 0;
  _first := false
}
@total := @total + msglen;
if (_last) status("Total length of messages = ",@total);
1;
```

15.11. Filter Troubleshooting

There are a number of common mistakes, particularly with the more complicated filters, which can often lead to unexpected results when using filters. Check your filter against the list below if you see odd behaviour.

- ### Did you remember to return a Boolean ?
  Remember that as well as processing some data the filter must also return a Boolean value indicating whether the entry should be shown or not. In a number of cases above this is achieved simply by adding the characters **;1** to the end of the filter. This ensures that the filter always returns true and therefore the entry will always be displayed.

- ### Are you using files ?
  Remember that a '\' character must be represented by '\\' in a file name.

- ### Remember to use := not = for assignment
  If you're using user variables then you must use := to assign a value. Both x := 3 and x = 3 are syntactically correct but the first is assigning the value 3 to x and the second is comparing the value of x with 3. Make sure you use the right one.

- ### A displayed user variable and one that isn't aren't the same variable
  The act of displaying a variable actually changes where the variable is stored which therefore changes the value. In other words @x# (which is displayed on the screen) is not the same variable as @x. You can prove this with simple test filters

  - **@x# := 5; @x := 2 ; 1**
    This filter will generate a column for **x** all containing 5

  - **@x# := @x# + 1; 1**
    This filter shows a column of all 1's. Why do we not see the number increasing for each entry ? Well, the displayed field is set to zero initially for each entry.

    If we wanted to see an increasing number we'd have to use the following filter.

  - **@x := @x +1; @x# := @x; 1**
    Here we see different numbers and they seem to be increasing but there are not in order. The reason for this is that the order displayed in the dialog is not necessarily the order that the objects are returned from the command server.

- ### Remember to switch _first off
  It is very common to want to do some initialization in your filter like reset counters to zero. To do this you put it inside an **if (_first)** block. However, remember that you yourself must set **_first** to false. If you don't your counters will be reset to zero for every entry in the list.

- ### Remember operator precedence
  For example, suppose you wish to see the messages where reply queues don't match the pattern "*IMS*". At first glance you might think is was **! replyq == "*IMS*"**. However, because the **!** operator is higher in precedence than == it will be executed first which will give a completely different result. The solution is to use brackets. The expression **! (replyq == "*IMS*")** will work fine.

Chapter 16. User Format Messages

MQEdit implicitly knows about the standard MQ message formats such as PCF, MQXQH, MQDLH etc. However, it is often the case that developers define their own message formats. Naturally MQEdit doesn't automatically know about these formats but you can provide descriptions of these message formats which then allows MQEdit to both display and edit the message in a more meaningful way than just hex. You can also use a User Format definition file to associate a particular formatter with a particular queue or format. For example you can say that a queue, say CUST.DATA, should, by default, use the JSON formatter since messages on that queue are normally in JSON format.

In the User Formats tab on the Preferences Dialog on page 113 you can specify the name of file which contains the definitions of the user message formats. If required this file can include other files so it is not necessary to put all the definitions in a single file but the definitions can be organised however is appropriate for your installation. For example, you could have a file per application group.

16.1. File Format

The file is a simple text file which can be created with any text editor. The file is largely free-format and white-space and comments can be places anywhere it is deemed appropriate. The one or two exceptions to this will be highlighted below.

By default MQEdit will look for a file called MQEDIT.FMT. However this can be changed in the User Formats tab on the Preferences Dialog on page 113 if required.

16.2. Comments

Any data following a ';' (semicolon) is regarded as a comment. These can be placed anywhere in the file. The comment marks the rest of the line as a comment.

```
; This comment is the entire line
struct Test
{
  int Field; Field comment
}
```

In most programming languages the comment is completely ignored by the parser. However, in MQEdit some comments are treated as part of the data. These are:

- If a comment immediately follows the structure name
- If a comment immediately follows a field definition
- If a comment immediately follows a n enumeration definition

16.3. Including other files

The definition file may, at any time, include additional definition files provided the keyword 'include' is at the top level. For example, you can not use **include** in the middle of a structure or enumeration definition. MQEdit will parse the included file just as though the text were entered in the original file. The syntax is as follows:

```
include “mydefinitions.fmt”
```

The file can be fully qualified with the full path if required. The the full path is not specified then the path of the current file is assumed.

There are many reasons why organising the definitions into different files may be useful but perhaps one of the major reasons is that the definitions may not all be coming from the same person. For example, consider a company which has, say, three application groups. We'll call them X,Y and Z. Each of these application groups is responsible the definition and structure of their MQ messages. One might imagine that they are also responsible for producing an MQEdit file containing the definitions suitable for browsing and editing the messages. So, we could imagine that the MQ administrator may get three files:

- **MQEdit_X.fmt**
- **MQEdit_Y.fmt**
- **MQEdit_Z.fmt**

Clearly the administrator could try to merge these definitions into a single file. However, this is time consuming and error prone. Instead it makes more sense to define a single fixed file and include the application group files.

One further reason why having each structure in a different file is the helper program *MakeFmt. MakeFmt* is free program supplied by MQGem Software that will convert a COBOL cobybook into a user format definition. This can make it make much easier and quicker to generate a representation of the COBOL structure in the User Format syntax.

For example, the content of **MQEdit.fmt** may be:

```
; Include application group definitions
include "MQEdit.X.fmt"
include "MQEdit.Y.fmt"
include "MQEdit.Z.fmt"

; Additional common definitions
struct XXX
{
  …......
}
```

Now when an application group produces a new set of definitions the administrator merely needs to copy the new file over the top of the old one. To use the definitions it is not even necessary to end the editor but just go in to the User Formats tab of the Preferences Dialog and press the "Reload User Formats" button.

16.4. File Syntax

There are essentially three aspects to the definitions.

1. **Structure Definitions**
   These definitions define the data structures and the fields that are contained in them.
2. **Enumerations**
   These definitions allow you to specify special values for a field and given them a name.
3. **Structure Association**
   These definitions allow you to associate either an MQ format or MQ Queue name (or both) with a particular structure definition.

We will discuss these in turn.

16.5. Structures

These definitions are the definitions which actually layout the format of the user data. Each structure definition must have a unique name. Structures contain one or more field definitions, and can also contain other structure definitions.

The general syntax of the definition is:

```
struct <Name> [ ; Optional Description ]
{
  ; As many field definitions as required
  <Type> [<Type Format>]
         [<Type Name>]
         [<Field Name>]
         ['[' <array value> ']']
         ['(' <enumeration> ')']
         ['=' <conditional>] [':' <option list> ] ';'
  . . . .
}
```

At first glance this may appear complicated but it is actually pretty simple. The general idea is that the definition looks very similar to the syntax we might find in any number of programming languages so it should be fairly natural to most technically minded people. Let's look at the simple example:

```
struct CUSTOMER; Customer Record
{
  char ID[4]      ; Eyecatcher
  int  Version;
  char CusName[50]; Customer name
  hex  AuthToken[16];
}
```

Here we can see a simple structure containing just a few fields looking a little like a 'C' structure definition. However, there are a few key differences. Firstly the field types are different than those defined in 'C' and far more varied. For a complete list of the field types please see 'Field Types' below. Another key difference is syntax and usage of comments. As mentioned in the ' Comments' section a comment can be considered as data. For example the comment “Customer Record” will be used as the description of the structure in the message display. If this comment is not present then MQEdit will use the structure name, CUSTOMER, as the structure description. In a similar way the comments, “Eyecatcher” and “Customer Name” will also be taken as the values which should be displayed for the field. The key difference for a field comment is that only the first 14 characters will be used.

So, if we used this structure definition and display a message we will get output like the following:

```
[   74 bytes] Customer Record
Eyecatcher   :'CUST'
Version      :1
Customer name:'MQGem Software Limited                            '
AuthToken    :01234567890123456789012345678901
```

We can see that MQEdit effectively overlays the defined structure on to the message data. One of the other aspects that this example demonstrates is one of the uses of arrays. Here we are using arrays on the ID, CusName and AuthToken fields to specify that the field consists of a fixed number of the primitive element types. So, for example, CusName is a fixed string of 50 characters. For a complete description of arrays please see 'Field Arrays' below.

Some of the definition elements are only applicable to certain types of fields. For example,

16.5.1. Structure Descriptions

In addition to the field definitions you can also request MQEdit output description lines in the structures. This is done by using a type value of '**desc**'.

The syntax for a description line is as follows:

```
struct <Name> [ ';' < Description Text > ]
{
  desc [':' <option list> ] ';' < Description Text >
}
```

A structure definition can have as many description lines as you wish. The description text itself is optional and so these lines can be used to just space out the fields.

The first blank character of the description text, if present is ignored. So, if you wish to actually indent the description in the output structure by 4 spaces you must include 5 spaces in the definition.

If we added a description line to the structure above we could do it with this definition:

```
struct CUSTOMER; Customer Record
{
  char ID[4]      ; Eyecatcher
  int  Version;
  desc ; Identification Fields
  char CusName[50]; Customer name
  hex  AuthToken[16];
}
```

This would yield an output such as:

```
[   74 bytes] Customer Record
Eyecatcher   :'CUST'
Version      :1
Identification Fields
Customer name:'MQGem Software Limited                            '
AuthToken    :01234567890123456789012345678901
```

Clearly many of the field options are not applicable to descriptions lines. However, using **low**, **medium** and **high** to control when the description line is displayed.

16.5.2. Field Types

MQEdit supports many field types, far more than you might expect to find in a normal programming language. This allows you to control how the data is overlaid into the structure and how the data is displayed to the user. The full list of data types is given below:

Data Type	Size	Description	Example Output
align16	Variable	Align next field to 16 byte boundary	None
align2	Variable	Align next field to 2 byte boundary	None
align4	Variable	Align next field to 4 byte boundary	None
align8	Variable	Align next field to 8 byte boundary	None
char	1 byte	Character in current codepage	`Field Value  :'X'`
charbp	n bytes	Blank padded character in current codepage	`Field Value  :'X   '`
desc	0	Prints out text line in structure	`This can describe the fields in the structure`

digits(n)
digits(m.n)
	Variable	Signed Decimal character fields
		For example:
		digits(5)
		digits(3.2)
		See 'Signed Decimal Character Fields' for more information.
			`Field Value  :-12345`
			`Field Value  :+123.45`

double	8 bytes	8 byte floating point number See 'Floating Point Numbers' for more information	`Field Value  :23.356`

flags	4 bytes	Integer which represents a set of flags	`Field Value  :00000103`
			`00000001 Flag 1`
			`00000002 Flag 2`
			`00000100 Flag 8`

float	4 bytes	4 byte floating point number See 'Floating Point Numbers' for more information	`Field Value  :23.356`
hex	1 byte	Hexadecimal byte	`Field Value  :7C`
int	4 bytes	32bit signed integer	`Field Value  :37`
int64	8 bytes	64bit signed integer	`Field Value  :2323423423423423423`
int64h	8 bytes	64bit signed integer, with following hex	`Field Value  :2323423423423423423 [0x'203E754B27F9A3BF']`
int8	1 byte	8bit signed integer	`Field Value  :37`
int8h	1 byte	8bit signed integer, with following hex	`Field Value  :37 [0x'25']`
inth	4 bytes	32bit signed integer, with following hex	`Field Value  :37 [0x'25']`

opunch(n)
opunch(m.n)
	Variable	Over-punched signed Decimal
		For example:
		opunch(5)
		opunch(3.2)
		See 'Over-punched Signed Decimal Fields' for more information.
			`Field Value  :-12345`
			`Field Value  :+123.45`

packed(*n*)	Variable	Signed Packed Decimal
		For example:
		packed(6)
		packed(5.3)
		See Packed Decimal Fields for more information.
			`Field Value  :-123`
			`Field Value  :12.340`

short	2 bytes	16bit signed integer	`Field Value  :37`
shorth	2 bytes	16bit signed integer, with following hex	`Field Value  :37 [0x'25']`
struct	Any	Embedded structure	As per structure content
time32	4 bytes	4 byte time value Number of second since 00:00:00 January 1st 1970	`Field Value  :14:23:17 05/08/1997 (Local)`
time64	8 bytes	8 byte time value Number of second since 00:00:00 January 1st 1970	`Field Value  :14:23:17 05/08/1997 (Local)`
uint	4 bytes	32bit unsigned integer	`Field Value  :37`
uint64	8 bytes	64bit unsigned integer	`Field Value  :2323423423423423423`
uint64h	8 bytes	64bit unsigned integer, with following hex	`Field Value  :2323423423423423423 [0x'203E754B27F9A3BF']`
uint8	1 byte	8bit unsigned integer	`Field Value  :37`
uint8h	1 byte	8bit unsigned integer, with following hex	`Field Value  :37 [0x'25']`
uinth	4 bytes	32bit unsigned integer, with following hex	`Field Value  :37 [0x'25']`

upacked(*n*)	Variable	Unsigned Packed Decimal
		For example:
		upacked(6)
		upacked(5.3)
		See Packed Decimal Fields for more information.
			`Field Value  :123`
			`Field Value  :12.340`

ushort	2 bytes	16bit unsigned integer	`Field Value  :37`
ushorth	2 bytes	16bit unsigned integer, with following hex	`Field Value  :37 [0x'25']`

16.5.2.1. Over-punched Signed Decimal Fields

Over-punched Signed Decimal fields are numeric fields where each digit of the number is stored in a single character. The physical size of such a field depends on how many digits it needs to support. The leading or trailing byte (depending on the definition) will contain some bits that represent the sign. For full details of the format of Over-punched ASCII fields please read Appendix A: Over-punched ASCII format on page 148.

The definition of this field can take two forms and both forms have some qualifiers defined below.

1. **opunch(a,*qualifier*) – an integer value**
2. **opunch(a.b,*qualifier*) – a real number value**

where:-

- **a** gives the number of digits before the decimal point or number of digits in an integer.
- **b** gives the number of digits after the decimal point.
- ***qualifier*** can take one (or more) of the values in the table in 16.5.3. Type Format on page 79. Multiple values are separated with the '|' sign.

For example, opunch(5) would allow numbers in the range -99999 to 99999 and opunch(3.2) would allow numbers such as +123.40

16.5.2.2. Packed Decimal Fields

Packed Decimal Fields are numeric fields popularized, and used heavily, by the COBOL programming language. Packed Decimal fields are numeric fields where each digit of the number is stored in a single nibble. So, unlike a data type, such as int, where the physical size of the bytes is fixed the size of a packed field depends on how many digits it needs to support[16]. The definition of the packed or unpacked field can take two forms.

1. **packed(a) – an integer value**
   where **a** gives the number of digits to support.
   For example, packed(5) would allow numbers in the range -99999 to 99999.

   The editor will not display leading zeros of the number.
2. **packed(a.b) – a real number value**
   where **a** gives the number of digits before the decimal point
   where **b** gives the number of digits after the decimal point
   For example, packed(3.2) would allow numbers such as 123.40

   The editor will not show leading zeros but it will show trailing zeros as above.

16.5.2.3. Signed Decimal Character Fields

Signed Decimal Character fields are numeric fields where each digit of the umber is stored in a single character. The physical size of such a field depends on how many digits it needs to support and additionally it has one separate character to store the sign symbol (a '+' plus or a '-' minus sign). The definition of this field can take two forms and both forms have some qualifiers defined below.

1. **digits(a,*qualifier*) – an integer value**
2. **digits(a.b,*qualifier*) – a real number value**

where:-

- **a** gives the number of digits before the decimal point or number of digits in an integer.
- **b** gives the number of digits after the decimal point.
- ***qualifier*** can take one (or more) of the values in the table in 16.5.3. Type Format on page 79. Multiple values are separated with the '|' sign.

For example, digits(5) would allow numbers in the range -99999 to +99999 and digits(3.2) would allow numbers such as +123.40

16.5.2.4. COBOL equivalents

If you are using packed fields in your messages then there is a good chance that you are also using COBOL. The following table shows COBOL data types and their equivalent User Format type.

COBOL Data Type	MQEdit User Format Data Type
`PIC S9(n) COMP-3`	`packed(n)`
`PIC 9(n) COMP-3`	`upacked(n)`
`PIC S9(n)V99 COMP-3`	`packed(n.2)`
`PIC S9(n)V9(m) COMP-3`	`packed(n.m)`
`PIC S9V99 COMP-3`	`packed(1.2)`
`PIC S9(n) SIGN LEADING`	`opunch(n, signleft)`
`PIC S9(n) SIGN TRAILING`	`opunch(n, signright)`
`PIC S9(n) SIGN LEADING SEPARATE`	`digits(n, signleft)`
`PIC S9(n) SIGN TRAILING SEPARATE`	`digits(n, signright)`
`PIC S9(n)V99`	`opunch(n.2)`
`PIC S9(n)V9(m)`	`opunch(n.m)`
`PIC S9V99 SIGN TRAILING SEPARATE`	`digits(1.2)`

---

[16] COBOL is heavily used in back-end systems such as z/OS running on s390 hardware and most information seems to pertain to that system. So much so that we could not find reliable information about how packed decimal fields are stored on, say, an Intel machine. As such MQEdit will always treat packed decimal fields as big endian format. If anyone can provide us with the nibble layout of a packed decimal field on other platforms then we'd be happy to also support those.

16.5.3. Type Format

This field can modify the way in which a particular field type is processed and displayed. It is only applicable to certain types of fields.

The definitions are:

Data Type	Type Format	Description	Example Definition
double	<Precision>	How many significant digits are output. Default:16	`double(13) Value;`
float	<Precision>	How many significant digits are output. Default: 7	`float(5) Value;`
time32	Comma separated list of options		
	UTC	Display time in UTC rather than Local time	`time32(UTC)          StartTime;`
	MMDDYYYY	Display date in MM/DD/YYYY rather than the default DD/MM/YYYY	`time32(UTC,MMDDYYYY) StartTime;`
time64	Comma separated list of options		
	UTC	Display time in UTC rather than Local time	`time32(UTC)          StartTime;`
	MMDDYYYY	Display date in MM/DD/YYYY rather than the default DD/MM/YYYY	`time32(UTC,MMDDYYYY) StartTime;`

char
packed
upacked
	Please see Field Description for a list of possible values.	This feature allows you to give MQEdit information about the content of the field which is then conveyed to the user. For example you can indicate that this is a date field. The user would then be presented with additional information to allow them to more easily understand the field value.	`char(YYMMDD)         Birthday[6];`
			`packed(8,DDMMYYYY)   OrderDate;`

packed
upacked
opunch
digits
	lead	Display leading zeros for the number	`packed(8,lead,DDMMYYYY)  OrderDate;`
			`opunch(3.2,lead)         Price;`
			`digits(5,lead)           Total;`

opunch
digits
	signleft	For an `opunch` type this means the number contains an over-punched sign on the LHS.
		For a `digits` type this means the number uses a separate extra character to store the sign on the LHS.
			`opunch(3.2,signleft)   Price;`
			`digits(5,signleft)     Total;`

opunch
digits
	signright	For an `opunch` type this means the number contains an over-punched sign on the RHS.
		For a `digits` type this means the number uses a separate extra character to store the sign on the RHS.
			`opunch(3.2,signright)   Price;`
			`digits(5,signright)     Total;`

opunch
digits
	plus	Display the plus sign when the number is positive (the minus sign is always shown when the number is negative).	`opunch(3.2,plus)       Price;`
			`digits(5,plus)         Total;`

16.5.4. Field Description

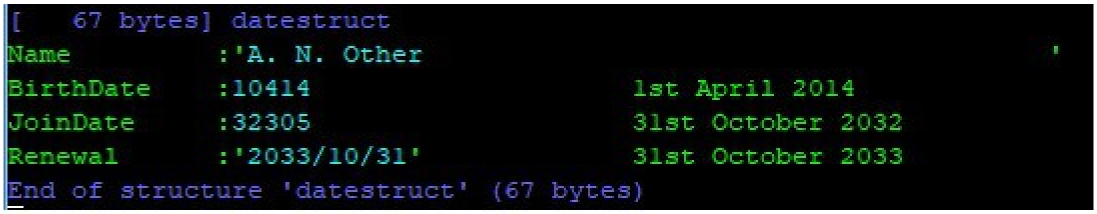
There are times where just looking at a field value may not be sufficient to easily interpret the data. Consider a date field where the value is '010203'. Does this mean 1st February 2003 or perhaps 3rd February 2001. Our American friends may see it as 2nd January 2003. What we need is some way to indicate that the field is actually DDMMYY and this is where a field description comes in. For character and packed fields you can optionally follow the type with a description string. A description string can consist of multiple elements, each one describing a character or digit of the field value.

Element	Meaning
DD	The two digit day of month (1-31)
hh	Hours
JJJ	Julian Day – day number from 1st January
mm	Minutes
MM	Month Number (1-12)
ss	Seconds
yy	A two digit year which is not interpreted – displayed as '84
YY	A two digit year where values less than 50 are considered 20YY. Values greater than 50 are 19YY For example 34 would be displayed as year 2034.
YYYY	A four digit year

MQEdit will look at the field value and its description and interpret the data and display the realised string on the right-hand side of the field. So, let's look at a couple of simple example field definitions. Suppose we had this set of definitions:

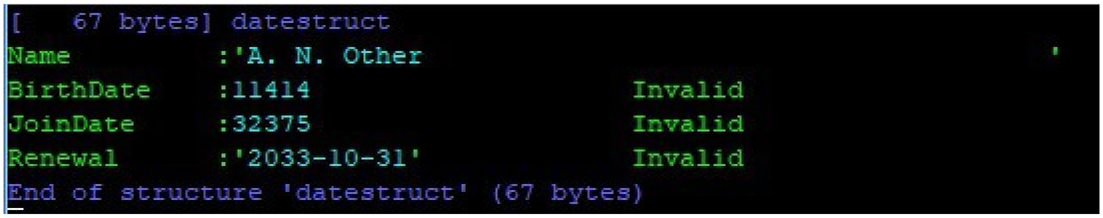
```
struct datestruct
{
  char                Name[50];
  packed(6,DDMMYY)    BirthDate;
  packed(5,YYJJJ)     JoinDate;
  char(YYYY/MM/DD)    Renewal[10];
}

queue DATES
{
  struct datestruct;
}
```

Clearly this is just an example and one would hope that no software designer wishing to show his face in polite society would ever define a structure with three date fields all of different formats, But here we are....this is the level of education we have today! Anyway, such a structure, if ever designed would be displayed as the following:



```
[   67 bytes] datestruct
Name          :'A. N. Other                                    '
BirthDate     :10414                    1st April 2014
JoinDate      :32305                    31st October 2032
Renewal       :'2033/10/31'             31st October 2033
End of structure 'datestruct' (67 bytes)
```

There are a couple of things worth noting. Firstly, note that the convention of not showing leading zeroes on a numerical field leads to a slightly odd look for the actual data field. We have a six digit date displayed as a five digit number. You can fix this using the 'lead' type format. However, the description on the right helps us confirm that the data entry is correct. Secondly, note that Julian dates are very difficult to read on their own. The JoinDate value of ***32305*** is not at all obvious as 31st October unless you have a mind like a steel trap. Now, suppose the value of the fields were incorrect? Well, we'd see something like this:



```
[   67 bytes] datestruct
Name          :'A. N. Other                                    '
BirthDate     :11414                    Invalid
JoinDate      :32375                    Invalid
Renewal       :'2033-10-31'             Invalid
End of structure 'datestruct' (67 bytes)
```

See if you can determine why each of these fields are now invalid. Displaying 'Invalid' can be a very useful way of indicating to the user that some of the message contains incorrect data.

16.5.5. Type Name

Type Name is only applicable for **struct** types.

```
struct MyStruct
```

The name is mandatory and gives the name of the structure definition that should be used for this field. This structure definition must already be defined. The structure name can optionally be followed by a field name.

16.5.6. Field Name

The name of the actual field. This field is mandatory for all field types except for the following types:

Data Type	Required
align2 **align4** **align8** **align16**	Should not be specified
struct	Can be optionally specified
desc	Should not be specified

16.5.7. Field Arrays

Field arrays are a way of telling MQEdit that a particular field repeats. There are two types of array.

16.5.7.1. Fixed Arrays

A fixed array, as it name suggests, has a fixed size. This is often used for character or hex fields but can be used for any data type. We have already seen arrays used in this way:

```
char CusName[50]; Customer name
```

Here we have defined that the customer name field is a fixed size of 50 characters.

However, suppose we want an array of names. Well, we can just add another array dimension.

```
char CusNames[50][10]; Customers names
```

This definition defines a 10 element array of 50 character names.

In a similar fashion we can define hex fields.

```
hex AuthenticationCode[32];
hex AuthenticationCodeArray[32][5];
```

16.5.7.2. Variable Arrays

A variable array is an array which can change size as the message content itself changes. This concept is only applicable to **char,** **hex** and **struct** type fields. There are essentially two forms. Firstly where the size of the array is given by the value of another field. For example, the following:

```
int  CusNameLen         ; Name Length
char CusName[CusNameLen]; Customer name
```

In this case the length of the customer name field is given by the **CusNameLen** field. Or you could do the following:

```
int    NumCusRecords            ; Number of records
struct CusRecords[NumCusRecords]; Customer name
```

where the number of following structures is given by a numeric field. Of course because you can have variable numbers of structures you can effectively have variable numbers of any field, it just depends on the definition of the structure.

Using variable length fields will mean that the offset of any fields following the variable array will change as its length changes. You may want to use a field option such as 'pad4' in order to align the following fields. Alternatively you can defined align fields into the structure definition to ensure that following fields are kept on an appropriate boundary.

The second form of variable array applies only to **char** and **hex** type fields which support a special value of '*' which can be used to indicate that the field applies to the rest of the message. Clearly a field using this syntax should be the last field defined in the structure and message.

For example:

```
char EyeCatcher[4];
int  Version;
char Details[*];
```

Here the **Details** field starts at offset 8 into the message and it's length is the remainder of the message.

16.5.8. Conditional Values

Conditional Values are a way of making the structure selection based on the content of the data. In other words a structure definition will only apply if all the conditional values match. Consider the example:

```
struct CUSTOMER; Customer Record
{
  char ID[4]      ; Eyecatcher
  int  Version;
  int  Type = 1;
  char CusName[50]; Customer name
  hex  AuthToken[16];
}
```

Here we see that a conditional value has been put on the **Type** field. If the underlying data does not have the value '1' in this position then MQEdit will not think this structure applies, In this way you can define multiple structure definitions and let the conditional values drive which structure is in operation.

Conditional values can only be placed on numeric fields, fixed string arrays and fixed hexadecimal arrays.

16.5.8.1. Conditional Strings

```
struct CUSTOMER; Customer Record
{
  char ID[4] = “CUST”; Eyecatcher
  int  Version;
  int  Type = 1;
  char CusName[50]; Customer name
  hex  AuthToken[16];
}
```

Here we are putting the conditional value on the character array. The conditional string can not be longer than the array field. If it is shorter then only the supplied characters are matched. Note that only characters that apply in both the source and target codepage can be used.

Note that this example has specified two conditional values. Both values must match in order for the structure to be considered a match.

16.5.8.2. Conditional Hex

```
struct CUSTOMER; Customer Record
{
  char ID[4]      ; Eyecatcher
  int  Version;
  hex  Type[4] = “AE45CFED”;
  char CusName[50]; Customer name
  hex  AuthToken[16];
}
```

Here we are putting the conditional value on the hexadecimal array. The conditional string can not be longer than the array field. If it is shorter then only the supplied values are matched.

16.5.9. Field Options

Each field can optionally have a list of options values which control the behaviour of the field and its output.

The list of options and their meanings are as follows:

Option	Meaning
align4	Tells MQEdit to always assign storage to this field in multiples of 4 bytes
ccsid	This field will set the codepage value for the following chained structure
defer	Editing of this field is deferred. The messages byes are not changed until the user presses <ENTER>
enc	This field will set the encoding value for the following chained structure
expert	This field can only be modified in expert mode.
high	This field is always displayed
limval	Limited Values This option is used in conjunction with enumerations. It tells MQEdit that this field only has a limited number of defined values and that any unrecognised values should be ignored. If this option is not specified then MQEdit will display **(Unrecognised)** next to to the field
low	This field should be displayed in low detail mode and above This is the default if no detail level option is specified
medium	This field should be displayed in medium detail mode and above
struclen	Identifies the field as the one containing the length of the structure
summary	Identifies this field as one that should be shown in the message list pane.

16.6. Enumerations

Enumerations are useful for a number of reasons.

1. To display a text string description for a 'known' field value
2. To list the bit values which should be displayed for the flag fields.
3. To allow simple data entry of a 'known' value just by double clicking on the value.
4. To display the list of possible values to the user.

There are actually two types of enumeration definitions and which one to use depends on the type of field.

16.6.1. Integer Enumeration

This type of enumeration must be used for any type of field which stores an integer number in binary form. In other words the following field types **int**, **short**, **flags**, **uinth** etc.

The format of the enumeration is the same whether you are displaying a single version or a set of flags. The key difference is that when an enumeration is used for a flag field then the enumeration values should be powers of 2. Let's look at a simple numeric enumeration.

```
enum_int Values
{
  1,  "VAL1" ; First Value
  2,  "VAL2" ;
  3,  "VAL3" ; Third Value
  4,  "VAL4" ; Fourth Value
}
```

Each enumeration value consists of three parts. The actual value constant, the constant name and the constant description. Specifying a constant description is optional. So, while the ';' (semicolon) at the end of each value is mandatory the text following the ';' is optional. If no description is specified then the description is assigned the same value as the constant name.

This enumeration can be used in a structure with the following definition:

```
struct EXAMPLE ; Numeric Enumeration Example
{
  int  Field(Values); Main field
}
```

The enumeration must be defined before it is used.

An example of a flag enumeration would be:

```
enum_int ValueFlags
{
  0x00000001,  "Flag1" ; Flag One
  0x00000002,  "Flag2" ; Flag Two
  0x00000004,  "Flag3" ; Flag Three
  0x00000008,  "Flag4" ; Flag Four
}

struct EXAMPLE ; Flag Enumeration Example
{
  flags Field(ValueFlags); Main field
}
```

Any flags in the **Field** value which are not catered for by the enumeration flags will be output as a single additional value.

16.6.2. Character Enumeration

The second type of enumerations is called character because the 'value' of the enumeration is given by a character value rather than just an integer number. However, this type of enumeration is still used for some numeric fields. The syntax and a use of a character enumeration is very similar to an integer enumeration.

```
enum Countries
{
  "AU",  "Australia";
  "CA",  "Canada";
  "DE",  "Germany";
  "DK",  "Denmark";
  "FR",  "France";
  "NL",  "Netherlands";
  "NZ",  "New Zealand";
  "SE",  "Sweden";
  "UK",  "United Kingdom";
  "US",  "United States";
  "ZA",  "South Africa";
}

struct EXAMPLE ; Character Enumeration Example
{
  char CountryCode[2](Countries);
}
```

In the example above we can see a two character field containing a country code which 'points' to a character enumeration containing the country codes and their text descriptions[17].

Character enumeration isn't only used for character fields however. They are also used for **packed**, **opunch**, **digits**, **float** and **double** fields.

```
enum InterestingNumbers
{
  "0",    "Zero";
  "0.58", "Euler's constant";
  "1.62", "Golden Ratio";
  "2.72", "e – natural log base";
  "3",    "First odd prime";
  "3.14", "pi";
  "13",   "Unlucky for some";
  "27",   "First odd perfect cube";
  "51",   "First uninteresting number!";
  "69",   "Square and Cube give complete set of digits";
}

struct EXAMPLE ; Character Enumeration Example
{
  float Value(InterestingNumbers);
}
```

When using an enumeration for numeric fields the value does not need to match exactly – for example a value of “1.00” would match “1”. As a general rule any trailing or leading numbers are ignored in the comparison. As a consequence it is entirely valid to just specify an integer value for double or packed(7.2) fields.

---

[17] Apologies to citizens of the hundreds of countries which are not represented here. Clearly in a real implementation the list of countries would be far larger and undoubtedly include yours. This subset was chosen purely for the sake of brevity and because it represents the majority of our customers.

16.7. Structure Association

In order to get MQEdit to use a defined structure you need to associate the structure with a particular Queue or Format. Traditionally it is the MQMD FORMAT field which tells MQ what the structure of the MQ message is. So, in an ideal world, you would have a unique FORMAT value for each of your structures. However, this approach has one major drawback. That is that in order to do data conversion you would need data conversion exits for each of your structures. It is possible, if you are just using character data in your messages, that a format of MQSTR would be sufficient and the context determines how the message should be treated. For example, you could say that all messages on queue 'CUSTOMER' contain only customer record messages.

For this reason MQEdit allows you to associate defined structures to particular format values or particular queue names, or both. The syntax is as follows:

```
association list   := format association | queue association | structure list
formatter name     := AUTO | TEXT | JSON | FIX | CSV | XML | EDIFACT

format association := format <format list> [ ';' < Description Text > ]
                      {
                        [ override section ]
                        [ formatter <formatter name>; ]
                        { association list }
                      }

queue association  := queue <queue list>
                      {
                        [ formatter <formatter name>; ]
                        { association list }
                      }

structure list     := <structure list> ;
```

Note that the definition is recursive, for example it is possible to include a queue association inside a queue association.

The format and queue lists can contain the standard wildcards * and ?. The format list also supports the special value **$none** which equates to an empty format field (MQFMT_NONE).

So, an example definition could be:

```
format CUS*,CST* ; Customer Record
{
  struct CUSTOMER;
}

queue CUSTOMER
{
  format $none, MQSTR
  {
    struct CUSTOMER;
  }
}
```

These two simple definitions state that any format beginning with the characters 'CUS' or 'CST' should display the CUSTOMER structure. Equally any message with formats of either MQSTR or NONE on the queue called CUSTOMER should also display the messages as the CUSTOMER structure.

The description “Customer Record” following the ';' on the format line will be displayed to the right of the format field e.g in the message descriptor.

One word of warning about the syntax. The semicolon essentially serves two purposes. It marks the end of the structure list but it also marks the start of a comment. Any characters beyond it on the same line will be ignored. This means that if you try to compress the definitions into a single line, such as the following, it **WILL NOT** work:

```
format CUS*,CST* { struct CUSTOMER; }
```

This is because MQEdit will not see the terminating '}' bracket. You should keep your definitions on separate lines, this ensures it is syntactically correct and is also more readable.

The other form of association you can make is to associate a formatter with a queue or format name. For example, suppose we have a queue, CUST.DATA, that we know always contains messages which are in JSON format. It would make sense to associate that queue with that formatter rather than constantly having to set it in the editor menu. We can do that simply by the following lines in the user format file.

```
queue CUST.DATA
{
  formatter JSON;
}
```

16.8. Override Section

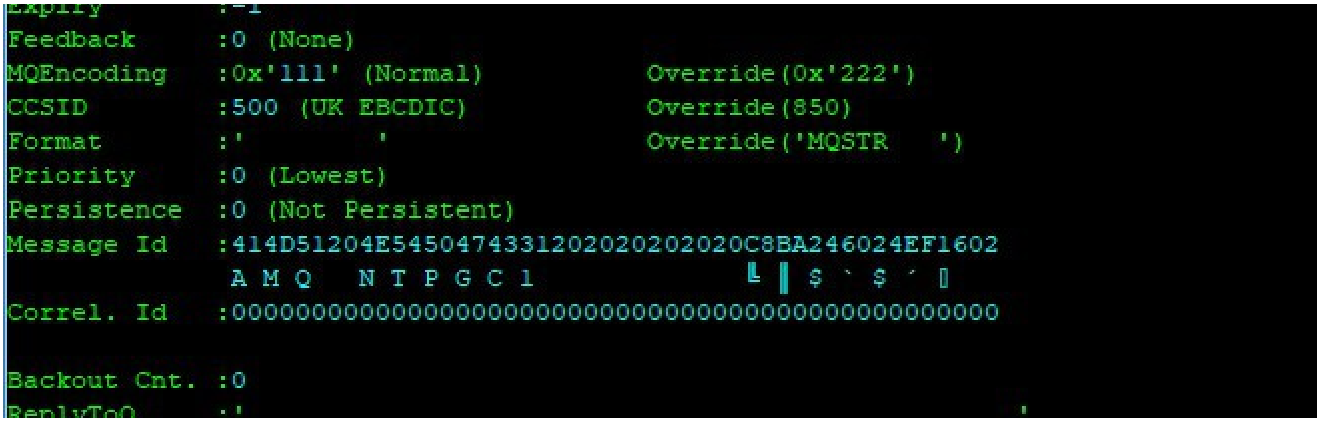
In an ideal world all MQ messages would contain all the correct identifying information regarding their content. Unfortunately, for a variety of reasons, that is not always be the case. We have seen MQ applications which have not used the correct format, codepage or encoding values. This means that a program like MQEdit may be looking at these fields in the Message Descriptor and making incorrect choices.

Of course, we would always recommend that MQ users go back and fix the applications putting these messages. Sometimes though, that is either not feasible or not possible. What we need is a way to tell MQEdit to ignore the values in the actual message and display the message as though a different set of values were in force. This is where the override section comes in. Let's look at an example:

```
queue Q1
{
  format *
  {
    override
    {
      CodedCharSetId = 850;
      Encoding       = 0x222;
      Format         = "MQSTR";
    }
  }
}
```

This definition states that every message on queue 'Q1', regardless of the given format value, should be treated as though it is a string format message in codepage 850 and with Intel-reverse encoding. Each of the override values are optional so you can override just one or two of the values if required.

If an override is in force then MQEdit will show the value when it displays the message. For example:



```
Feedback     :0  (None)
MQEncoding   :0x'111'  (Normal)          Override(0x'222')
CCSID        :500  (UK EBCDIC)           Override(850)
Format       :'        '                 Override('MQSTR   ')
Priority     :0  (Lowest)
Persistence  :0  (Not Persistent)
Message Id   :414D51204E5450474331202020202020C8BA246024EF1602
              A M Q   N T P G C 1               ╚ ║ $ ` $ ´ ▯
Correl. Id   :000000000000000000000000000000000000000000000000

Backout Cnt. :0
```

Here you can see that the editor shows you the actual message values, and allows you to edit them, but it also tells you on the right-hand side that it is displaying the messages as though a different set of values were set.

16.9. Floating Point Numbers

Floating point numbers present a lot of challenges for an editor. The main problem is that floating point numbers are inexact and the inexactness is not predictable. Floating point numbers encode an enormous range of numbers in a relatively small number of bytes. As such there are 'gaps'. There are some numbers that just can't be stored exactly. One of the factors which can affect this is the precision by which the number is printed out. For example, by default for **float** types the editor uses a precision of 7. This means that the editor will try to output 7 significant digits for any **float** variable.

For example, consider the number 1.2. In a float variable this is actually stored as 1.20000005. Similarly 1.224 is stored as 1.22399998. As you can see these numbers are very close approximations to the actual numbers but, none the less, are different. If the number is just displayed in a report then the different is unlikely to matter. However, in an editor it very definitely does matter. If you type the string 1.224 and 1.22399998 is displayed you would have a very strange editing experience. Consequently the editor tries to mask these differences by detecting and correcting these rounding errors. However, it means that if you enter a number which looks like a rounding error then the editor will, incorrectly, correct it.

If you find that the editor will not accept a particular floating point number then you can try entering a number which is close.

16.9.1. Precision

One of the key aspects of floating point numbers is the notion of precision. When a number is output you can say how many significant digits you want to display. However, bear in mind that floating point variable can only support a certain number of significant digits because of the limited number of bits the number is stored in. Again it is not an exact science but roughly speaking a **float** can support up to 7 significant digits and a **double** can support up to 16. Float variables, by virtue of their smaller size, are far more prone to rounding errors. You can control how many significant digits you would like displayed by following the data type with the number of digits you want. For example:

```
float(4)   number1;
double(10) number2;
```

If no precision is specified then MQEdit will use the logical maximum values of 7 and 16 respectively.

16.9.2. Display and Entry Format

Not only can floating point numbers store a number with great accuracy but they can also store very large or very small numbers using the exponential form. The exponential form is a fixed format:

```
<number>'e' [ '+' | '-' ] <3-digit exponent>
```

So, the following are all valid examples of floating point numbers using the exponential format:

```
1.2e+022

-4.567e+010

1.345e-024
```

Leading and trailing zeroes are not allowed. The choice about whether normal format or exponential format is used is based on the number itself. The exponential format should be used if the number is:

- equal or greater than 10000000000 (10^{10})
- equal or less than -10000000000 (10^{10})
- smaller than 0.0001 (10^{-4})

16.9.3. Specification

MQEdit is able to display and edit floating point numbers (single and double precision) in both IEEE 754 and IBM S/390 Hexadecimal floating point (HFP) formats. MQEdit refers to the Message Descriptor Encoding field (or Encoding field in any chained headers) to determine how any floating point numbers it comes across should be treated:-

- MQENC_FLOAT_IEEE_NORMAL
- MQENC_FLOAT_IEEE_REVERSED
- MQENC_FLOAT_S390

If your message does not correctly describe the floating point format that is in use in the message, see the 'Override Section' description on page 87.

16.10. Developing your User Format Definitions

MQEdit makes it fairly easy to develop your User Formats.

The suggested process is as follows:

1. **Decide on the message you wish to define a format for and display it in the editor.**
   Presumably, for the moment at least it is displayed either as just text or as hex.

2. **Now, display the bring up the “User Formats“ tab of the Preferences Dialog.**
   Leave this dialog open while you work on your format definitions.

3. **Ensure that the editor is displaying offsets.**
   This is most easily done by pressing the (+0) tool in the toolbar. Offsets can be displayed either in hex or in decimal, choose whichever you prefer in the “General“ tab of the Preferences Dialog. Displaying offsets is really useful since the editor will show you the offsets of each field in the structure as you define your message format. You can then verify that the offset is indeed what you would expect for the given data types.

4. **Next start a text editor and edit the format definition file**
   The name of the file will be displayed in the “User Formats“ tab. You can change the name to refer to any file you wish.

5. **To start with just define the first field of the structure you want MQEdit to display.**
   If you have a COBOL copybook then you can get a huge head-start with your strcuture definition by using the *MakeFmt* helper program.

   For example:-

   ```
   struct CUST;
   {
     char  CustName[50]; Customer name
   }
   ```

6. **Add an appropriate “Structure Association“**
   This tells MQEdit to use this structure definition for this MQ format or Queue. For example, to use a format of “CUSTOMER” we might specify the following:

   ```
   format CUSTOMER
   {
     struct CUST;
   }
   ```

7. **If necessary add conditional values to the structure definition.**
   Ideally, conditional values are added at the end of the process, once all fields are defined but it may be necessary to define them now to differentiate this structure from other structures.

8. **Press 'Save file' in the editor**

9. **Press the “Reload User Formats“ in the Preferences Dialog**

10. **MQEdit will display an error message if parsing the structure format file encountered any errors.**
    Continue to fix any syntax errors and pressing “Reload User Formats“ until parsing is successful.

11. **Now add the definition of the next field of the structure**
    If you don't have any more fields to add then you are done! If you do add a new definition then continue to step 8.

16.11. Creating a new user format message

MQEdit provides a number of ways of creating a message. For example, you could just go in to HEX mode and enter as many bytes as required. While flexible and exacting this can be a little laborious and error prone. Another approach might be to find a message which is 'close' to the one you want and modify the contents.

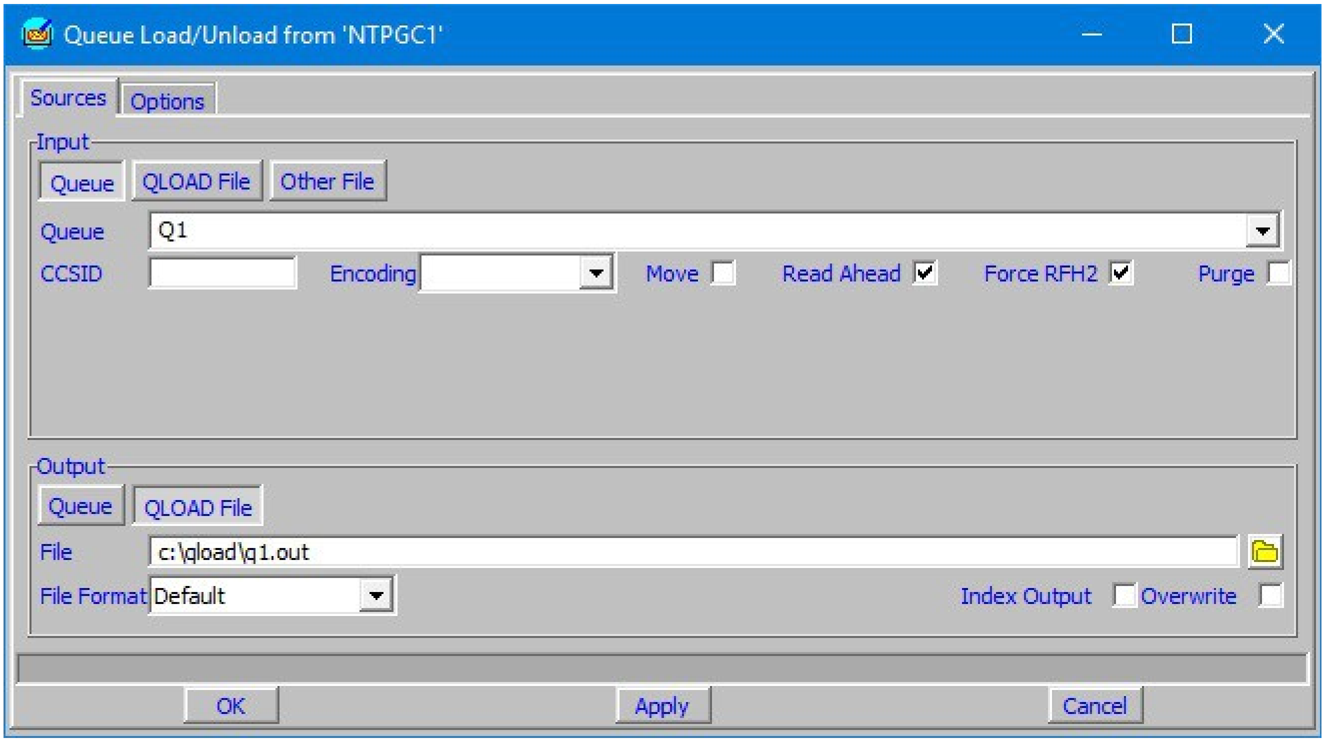
The simplest approach though is usually to use the 'New' message tool. 'New' message will look at the content of the current Message Descriptor Format field and make a message with a matching structure. This does therefore mean that you must have you structure association values defined before using this method. However, assuming that those definitions have been defined the 'New' message tool allows you to very quickly create messages of all of your user formats.

For a description of the 'New' message action please see “Creating a 'new' message“ on page 20.

Chapter 17. Queue load/unload facility

From various places, such as the message list pane, a menu option is provided which will present the queue load/unload dialog. This allows the user to unload messages from a queue and put them in a file, copy messages from one queue to another, or load messages from a file to a queue. This can be useful for a variety of reasons such as backing up your queues, saving a set of messages for later re-use, unloading a queue so that you can do some editing of the message or message attributes or for sending someone else a copy of your messages. If the operation takes more than a few moments then a Progress Dialog is shown giving an idea of when the operation will complete.

The load/unload facility uses the same file format as is used in our QLOAD command line queue load/unload product. As a consequence unloaded files can be shared between them.



Queue: Q1

CCSID:

Encoding:

Move ☐

Read Ahead ☑

Force RFH2 ☑

Purge ☐

File: c:\qload\q1.out

File Format: Default

Index Output ☐

Overwrite ☐

The dialog is split in two halves. For both input and output a choice has to be made about whether to use a file or a queue. So, for example, if the user chose an input of a queue and an output of a file then this would move messages from a queue to a file. In other words, it would unload the queue to the file. Note that it will not destructively get the messages; rather a copy of the messages will be taken unless the options 'Move' or 'Purge' are selected. If the input is a file then it can either be a standard QLOAD file or it can e an 'Other' file which is just a regular text file. Loading from an 'other' file allows you to load just a sequence of text records, please see 'Other File ' on page 94.

Both input and output can be of the same type. If they are both queues then you can do a copy of messages from one queue to another. If both are files then it allows you to reformat the output of the file. If the 'Auto Update browse lists' preference option is selected then any dialogs browsing the relevant queues will be refreshed after the queue load/unload operation.

The various options are:-

- **CCSID**
  If specified the messages will be 'got' from queue converted to the requested codepage.
- **Encoding**
  If specified the messages will be 'got' from the queue using the byte encoding requested.
- **Move**
  If this checkbox is checked then the messages will be removed from the source queue otherwise they will be copied.
- **Read Ahead**
  If this checkbox is checked then MQEdit will use a read-ahead handle if it can to retrieve the messages. Read ahead offers a significant performance advantage when used over client connections, particularly if those client connections are over slow network links. Read Ahead will only be active if either no message filtering is applied or messages are being copied rather than moved.

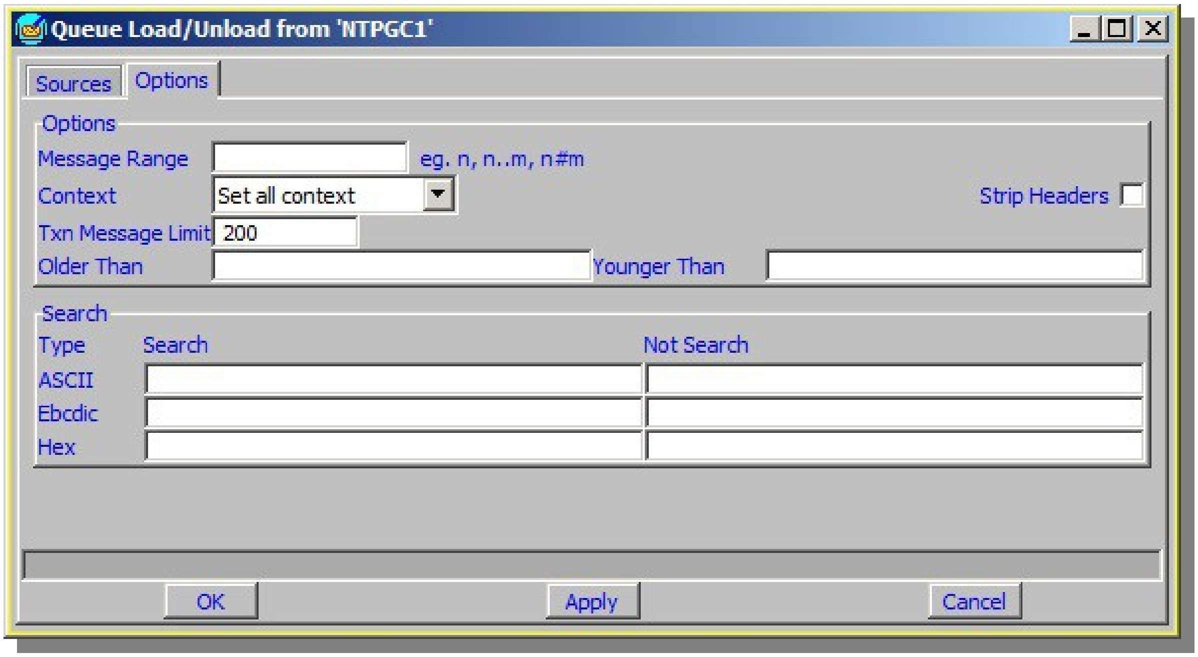
- **Force RFH2**
  This option ensure that if the message has any properties that they are written into the message body in the form of an MQRFH2 structure. If you want to ensure that all aspects of the message are stored you should check this option.
- **Purge**
  This option is only in effect when the move option is specified and some form of filtering is in effect in the options tab. When selected it says that messages not matching the selection should be removed from the queue (and are therefore lost since they **will not** be written to the target). If this option is not selected then messages not matching the selection will remain on the source queue.
- **Async. Put**
  If this checkbox is checked then MQEdit will use an Async repsonse put handle to put messages to a target queue. Async. Put offers a significant performance advantage when used over client connections, particularly if those client connections are over slow network links.
- **File Format**
  The following file formats are available:-
  - **Default**
    The message format is just a hex string output. This is the most compact format but not too friendly to the human eye.

    ```
    X 54657374204D657373616765
    ```
  - **ASCII**
    Message is presented in ASCII as much as possible. This is useful if the message contains a fair degree of ASCII which needs to be edited.

    ```
    S "Test Message"
    ```
  - **ASCII Column**
    Message is presented in hex but with a column of ASCII characters to the right-hand side. Any character in the message which can not be represented by a printable ASCII character will be displayed as a '.' (dot).

    ```
    X 54657374204D657373616765                  <Test Message>
    ```
- **Index Output**
  When selected the produced file will contain comments which indicate which message index each message is. This is useful when reloading only part of a file to identify each message by its index value.
- **Overwrite**
  By default the application will prompt the user if the output file already exists. If this option is checked then any existing file will be overwritten without requiring user confirmation. On the other tab a number of options are available to control which message are loaded/unloaded and what message attributes should be taken.

The options tab allows the setting of various values to control which messages are moved/copied.



Context: Set all context

Strip Headers ☐

Txn Message Limit: 200

- **Message Range**
  This option allows only a certain number of the messages to be processed. For example
  - **5** Would only process message number 5
  - **5..8** Would only process messages 5 through 8, inclusive.
    To process all messages from message 5 onwards enter a string like **5..999999**
  - **5#8** Would only process 8 messages starting with number 5
  - **#8** Would process the first 8 messages.
- **Context**
  Context setting which controls how the origin and identity context is transferred between the input and output source. The value of 'set all context' will ensure that all of the context fields are transferred and would therefore tend to be the most obvious choice. Note, however, that 'set all context' requires the most authority.
- **Strip Headers**
  This option controls whether headers, such as transmission queue and Dead Letter headers should be stripped before the message is moved or copied.
- **Transaction Message Limit**
  If you are moving messages from one queue to another then this must be done under a transaction to be totally reliable (in the case of persistent messages it is also much more efficient). However, there is a limit to how many operations you can perform in a single transaction so this value allows you to say how many messages form part of the transaction before the transaction is committed.
- **Older Than**
  This option allows selection based on the age of the message. The parameter can be entered free format using the keywords, **seconds, minutes, hours, days, weeks, years**. Only the initial parts of the keywords need be specified. For example, **1 hour 15 minutes** or **1 h 15 m** are equivalent.
- **Younger Than**
  This options allows selection based on the age of the message. The format of this parameter is the same as the **'Older Than'** field.
- **Search**
  You can choose to process only messages that contain a certain string. This string can be in ASCII, EBCDIC or hex. You can also do the opposite search - that is for messages that do not contain a certain string. You can use any combination of the search string options together. If more than one option is used, all search strings must match for the message to be processed.

17.1. Other File

Normally you would load messages from either another queue or from a QLOAD file. However, there are times when it is useful to be able to just a load text from a text file and construct messages accordingly.

The first setting to be chosen is whether the entire file should be treated as a single message or whether the file contains multiple messages. If 'single message' is selected then the entire contents of the file is read and a single message is written to the output queue or file.

However, if 'multiple messages' is selected then the file is parsed to determine the start and end of each messages. Clearly if the file contains multiple messages then there must be delimiters which mark the start and end of each message. You can specify these in the appropriate input fields. You can also specify whether those delimiters are included in the message text itself.

If no values are given then the default values used are NULL for the Start delimiter and newline for the End delimiter. This means that each line in the file will be treated as a new message. So the following file would yield three messages with contents 'One', 'Two' and 'Three'.

```
One
Two
Three
```

The newline is optional on the last line of the file.

However, suppose our messages have a little more structure to them. Let's consider the following file:

```
{"Message": {
  "customer": {
    "Name": "Fred","Address": "Somewhere"}
  }
}


{"Message": {
  "customer": {
    "Name": "Bob","Address": "Somewhere else"}
  }
}
```

Here we have a file containing two messages in JSON format. We wish to load these messages and have MQEdit recognise that there are two messages. All we need to do is set the start delimiter to '{' and the end delimiter to '}'. MQEdit will parse the file looking for the message start, '{'. The end of the message is given by '}'. However, MQEdit will not necessarily make the end of the message as the next '}' but will instead ensure that it is the matching bracket. MQEdit will match the following brackets : < > [ ] { } and ( ).

Since the message only contains the message text and no message descriptor values MQEdit has to construct those. Messages will always be put with a format of MQSTR. As for the message context this must be set (on the Options tab) to either 'No Context' or 'Default Context'.

When the user presses the 'OK' or 'Apply' button and the input is a non-QLOAD file then MQEdit will quickly check how many messages it finds in the file and present the user with a confirmation dialog. This is a useful check that there are indeed the number of messages in the file that you were expecting.

Delimiters do not need to be only be a single character but can be any sequence of characters. You could for example use HTML tags or just a sequence of your own choosing. For example [Message Start] and [Message End]. Whatever makes sense for your file.

Since we have delimiters at the start and end of the message it follows that some of the text in the file may fall out of the range of the message itself. In other words you can effectively have comments in your file. The following file will operate the same as the one above.

```
Comment for message one

{"Message": {
  "customer": {
    "Name": "Fred","Address": "Somewhere"}
  }
}

Comment for message two

{"Message": {
  "customer": {
    "Name": "Bob","Address": "Somewhere else"}
  }
}
```

This can be useful for documenting the purpose of a particular message in a set of test data.

17.2. QLOAD File Format

The format of the file that MQEdit uses is the same as our QLOAD product[18]. The file is deliberately human-readable to allow a user to update the file to easily make changes to the messages before loading them onto a queue, perhaps on a different system.

The file has a free format. Spaces and blank lines are ignored unless between quotes. When generated the file may contain spaces to make the file more readable, but they do not affect the message format stored in that file.

The first column contains the key for each line. This can have a number of different values, some of which we have seen already. These are listed in "Table 1: Meaning of column one symbol in file format".

Column 1 value	Meaning
`S`	The text shown on this line is ASCII string text
`X`	The text shown on this line is hex
`A`	The text shown on this line is an attributed in the Message Descriptor (MQMD)
`*`	The text on this line is a comment and will be ignored.

Table 1: Meaning of column one symbol in file format

The second column should contain a blank; this makes the file more readable.

If the first column contained the letter 'A', then the next three characters will represent the field that is being shown. For example, 'FMT' shows that this line displays the format of the message. The full set of these field labels is listed at the end of this section.

17.2.1. Example - Changing the user ID

The message descriptor (MQMD) contains a user ID which, depending on your security settings, may be used for access control when putting a message onto a queue. You may wish to change this user ID before reloading the messages onto a queue on a different system because the IDs on that system use a different naming convention. Before loading the queue from your file of messages, you would edit the file changing the field identified as USR to your desired user ID.

---

[18] The QLOAD product enables loading and unloading of IBM MQ Queues from a command line interface. This makes activities such as archiving queues easy in a command script.

```
:
A RTM MQ24
A USR HUGHSON
A ACC 1A0FD4D8F2F4C3C8C9D5F1F9C6F7C1C3F3F00019F7AC30000000000000000000
:
```

17.2.2. Attribute Format Reference

The fields in the Message Descriptor (MQMD) are formatted in the file with a three character string representing the attribute name. "Table 2: Message descriptor attribute representations" lists the full set of these strings and which field they represent.

Message Descriptor attribute	File format representation
Report	RPT
MsgType	MST
Expiry	EXP
Feedback	FBD
Encoding	ENC
CodedCharSetId	CCS
Format	FMT
Priority	PRI
Persistence	PER
MsgId	MSI
CorrelId	COI
BackoutCount	BOC
ReplyToQ	RTQ
ReplyToQMgr	RTM
UserIdentifier	USR
AccountingToken	ACC
ApplIdentityData	AID
PutApplType	PAT
PutApplName	PAN
PutDate	PTD
PutTime	PTT
ApplOriginData	AOD

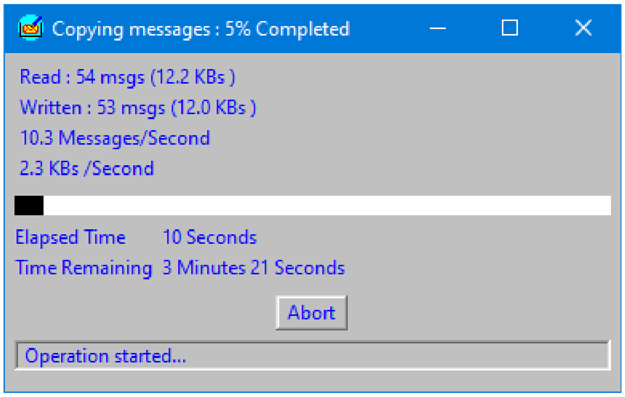
Table 2: Message descriptor attribute representations

17.2.3. Recognised file formats

MQEdit recognises the file format described above but also recognises the file format used as output from the sample browse program AMQSBCG. In other words, you can take an AMQSBCG output file and pass it as an input file to the load/unload dialog.

17.3. Progress Dialog

Copying, Moving, Loading, Unloading and Deleting large numbers of messages can take a considerable time. As such it is useful to get some feedback about how far along the operation has got. This is what progress dialogs are all about. By default the dialog is displayed if the operation takes longer than 3 seconds to complete. However, you can change this by setting the value of 'Progress Dialog Delay' in the Dialogs tab of the Preferences Dialog.



Copying messages : 5% Completed

Read : 54 msgs (12.2 KBs)
Written : 53 msgs (12.0 KBs)
10.3 Messages/Second
2.3 KBs /Second

Elapsed Time 10 Seconds
Time Remaining 3 Minutes 21 Seconds

Operation started...

As you can see the progress dialog will show you how many messages have been read and written so far. It also calculates the average number of messages per second which are being processed. Note that the average message speed counts both the input and output messages. MQEdit also tells you how many bytes per second we are moving.

This is followed by a visual representation of how far through the process MQEdit believes it is and an estimate on how long it will take. This is, of course, fraught with hazard. Any time any piece of software makes a prediction of how long something will take it opens itself up to complete ridicule! This is merely the best guess and is based on 'future messages being much like the ones I have seen'. As such the actual answer could vary wildly. For example the queue could be in use. If more messages are added or removed from the queue during the processing then clearly this will affect how long the operation takes. Another possible problem is varying message sizes. If the first few messages on the queue are small and these are followed by lots of 100 MB messages then again MQEdit may report an earlier completion than is possible. Needless to say if the messages get progressively smaller then in all likelihood MQEdit will complete faster than its initial estimate.

Finally, there is an 'Abort' button. Click this button if you wish the operation to stop. Exactly where it stops is, of course, not an exact science. If you are fast enough then perhaps nothing has happened. However, it is more likely that 'some' messages have been moved/copied. You may need to clean up these messages if you want to completely undo the operation.

As we mentioned before the 'Progress Dialog Delay' in the Dialogs tab of the Preferences Dialog controls whether the Progress Dialog is displayed. However, even if you choose not to display the Progress Dialogs for a long time that does not preclude you from displaying it if you wish. Progress Dialogs will be shown in the 'Windows' menu of the main window. By selecting the Progress Dialog in the Windows menu the Dialog will be shown. Similarly if you close the dialog and the operation is still continuing the dialog is merely hidden and can be redisplayed by selecting it again in the Windows menu.

17.4. Long Operations

MQEdit is not really designed for moving large numbers of messages around, not least of which because MQEdit may well be connected to the remote server over a client connection. Where possible you are probably better to use a local application, something like our QLOAD product. However, there is no denying that being able to just quickly move or copy a few messages in your message manipulation GUI is convenient, quick and easy. You need to bear in mind that MQEdit will use the same connection to the Queue Manager for the message move/copy as it does for other message editing. This means that until the move/copy operation is complete you will not be able to do other useful stuff, such as look at messages on another queue and edit those.

Most copy/move operations will complete in just a few seconds. However, if you find yourself regularly issuing copy/move commands which take a long time to complete it might be worth either running a different instance of MQEdit or creating another location to the same Queue Manager and using just that one for the copy/move operation. For instructions on how to create more than one connection to the same Queue Manager please see section 3.2.2. Creating two locations for the same Queue Manager on page 9.

Chapter 18. Main Menu

Here we describe the actions of all the menus available on the main window.

18.1. File

18.1.1. New Instance

This menu causes a new instance of the main window to be created. This second instance can be operated completely independently so you can edit two messages at once. You can drag&drop messages between instances[19].

18.1.2. Refresh Information

Causes the application to refresh its cached information about the selected Queue Manager. This can be used to update the information shown in the tree view in the Queue Manager navigation pane.

18.1.3. Open Location

Displays a dialog showing the values set for the location allowing the current configuration to be changed. See “Location Dialog” on page 116 for a description of the fields.

18.1.4. Copy Location

Displays a dialog to allow a new location to be added to the list. The dialog will be pre-filled with the values of the selected location. See “Location Dialog” on page 116 for a description of the fields.

18.1.5. Add Location

Displays a dialog to allow a new location to be added to the list. See “Location Dialog” on page 116 for a description of the fields.

18.1.6. Import Locations

Allows the user to have location definitions automatically created by importing the definition from an existing source.

- **From CCDT...**
  This option allows locations to be imported from either a binary or JSON CCDT file. The choice depends on the name of the file chosen. A JSON CCDT file should end with the **.json** file type. Any other file type will be assumed to be a binary CCDT file.

  Any client definitions that are found are presented in the dialog. An icon next to the Queue Manager name specifies the status of the definition.

Exclamation Mark (!)	Can not be imported because the Queue Manager name in the CCDT is blank.
No entry sign	Can not be imported because there is already a location definition for this Queue Manager.
Tick	Definition can be imported and is currently selected.
Cross	Definition can be imported but is not currently selected.

  If required you can set the queue manager group, the network names, and whether this location is MVS, using the fields provided below the list of queue managers.

  Pressing 'Import' will import those definitions which are shown with a 'Tick' against the name.

- **From MQEdit Configuration...**
  Allows the user to specify another MQEdit Configuration file to import locations from.

blank	Can be imported, there is no equivalent current definition.
Delta Symbol	Can be imported. There is a similar definition defined but it is different in some way.
No entry sign	No need to import this location because it is the same as a currently defined location.
Tick	Definition can be imported and is currently selected.

  Pressing 'Import' will import those definitions which are shown with a 'Tick' against the name.

  By default locations will use the same Queue Manager group as the source configuration file. However, if you can specify your own Queue Manager group name.

---

[19] Note that a new instance is different from invoking the MQEdit program again. Messages can not be dragged between different invocations of the MQEdit program.

- **From MQ Explorer File...**
  Allows the user to specify the XML file created when an export is issued from MQ Explorer. Any client connection information definitions that are found are presented in the dialog. An icon next to the Queue Manager name specifies the status of the definition.

No entry sign	Cannot be imported because there is already a location definition for this Queue Manager.
Tick	Definition can be imported and is currently selected.
Cross	Definition can be imported but is not currently selected.

  You may also choose whether the MQ Explorer Set names should be used as the MQEdit Group and Network Names.

  Pressing 'Import' will import those definitions which are shown with a 'Tick' against the name.

- **Local Queue Managers**
  Checks whether all of the local Queue Managers are defined in the MQEdit configuration. If one or more are missing a dialog is displayed which allows the user to automatically import a location definition.

18.1.7. Delete Location
Deletes the current selected location.

18.1.8. Save Configuration
The current configuration will be written to the configuration file, MQEDIT.CFG. The current configuration is automatically saved when the program ends normally.

18.1.9. Preferences
Displays a dialog to allow the user to change some global parameters on the way the program operates, such as saving dialog positions. See “Preferences Dialog” on page 105 for a description of the dialog.

18.1.10. About
The About dialog gives version and build date information. The dialog will also show licence information such as the userid and machine that the program is running on, who the licence is registered to and the licence expiry date.

18.1.11. Check for latest MQEdit version
Displays a dialog showing the current version of MQEdit in use and the latest available version ready for download. Clearly this function requires an internet connection. If the internet connection fails then a message box will be shown explaining the failure.

By default MQEdit will check for program updates periodically and display a dialog if a later version is available. This feature can be switched off if required in the preference dialog.

18.1.12. Help
Will try to display this manual. Note that the manual file should be placed in the same directory as the program or its location identified in the *‘Help file location’* preference field. Note that 'F1' will also display the manual.

18.1.13. Exit
Exits the program[20].
The current configuration will automatically be written to the configuration file.

18.2. Edit
The edit menu contains various actions specifically associated with the editing of a message.

18.2.1. Expert Mode
Some fields, in a structured message are hazardous to change, for example structure and string lengths. These fields can only be changed when in 'Expert Mode' to avoid inadvertently corrupting the message.

18.2.2. Undo
This menu option will undo the last change to the message.

---

[20] The first time the user requests to end the program it will disconnect from all the connected Queue Managers before shutting down. It is possible that this process will hang because, say, the network connection has been lost. Selecting ‘Exit’ again will end the program immediately bypassing the disconnect processing.

18.2.3. Redo
This menu option will redo the last undo.

18.2.4. Repeat Last Action
When modifying a structured message, such as adding a new array element or deleting a message structure, it can be useful to be able to easily repeat the operation.

18.2.5. Select All, Cut, Copy, Paste, Paste All
These actions allow you to cut, copy & paste data to/from the clipboard. In general the operations behave as you might expect but it is worth familiarising yourself with the section on “Select All, Cut, Copy, Paste, Paste All“ on page 38.

18.2.6. Expand Edit Panel
This menu action will expand the message edit panel to the whole window size. Selecting the menu again will restore the three pane view.

18.2.7. Find and Replace
This menu will bring up the 'Find and Replace Dialog'. Refer to that section on page 45 for a description of the dialog.

18.3. Action

18.3.1. Load/Unload Queue
Displays a dialog allowing the user to load/unload a queue from/to a file. See “Queue load/unload facility” on page 91 for more details on how to do this.

18.3.2. Filters
Displays the Filter Manager dialog. See “Filter Manager” on page 68 for a description of how to use this window.

18.3.3. Put Message
Will display a simple dialog for the selected location to allow a simple message to be put to a target queue. The message can only be a string (‘MQSTR’ format) but the user can choose how many messages are put and whether persistence or syncpoint is used. The user can also choose whether the message content comes from the dialog string or a file.

18.3.4. Publish Message
Very similar to the ‘Put Message’ above but in this case the message is published to the given Topic. The user can choose between publishing the message using the MQI directly or using queued publish depending on which TAB is selected.

18.3.5. Disconnect
Normally the application will disconnect from a Queue Manager based on the disconnect interval specified in the location dialog. However, if the user wishes to force disconnection earlier than this then this menu may be used.

18.3.6. Disconnect All Shown
This command will disconnect from all the locations shown in the list if they are currently connected.
You can use the main window search field to choose which locations this menu applies to.

18.3.7. Connect All Shown
This command will attempt to connect to all the locations shown in the list if they are not currently connected and it is appropriate. For example, MQEdit will not directly connect to locations which are processed 'via' another location. Connecting to a group of Queue Managers can be useful to make a quick check that the set of Queue Managers are running correctly and are available for connections. You can use the main window search field to choose which locations this menu applies to.

18.3.8. Local Error Log File
Selecting this option will display the contents of the local error log for this location. For example, if this location is accessed via an MQ Client then the client error is displayed.

18.3.9. Local INI file
Selecting this option will display the contents of the local INI file for this location. For example, if this location is accessed via an MQ Client the client INI file is displayed,

18.4. View

18.4.1. Set Font

There are two fonts which can be set within the program. One used for ‘Windows’ and one for displaying the message data.
The windows font is used as the font for the main window and the command windows.
The message font is just used as the font for displaying message content. The message display will always be presented in a non-proportional fashion to ensure columns of characters a aligned. However, the font itself can be either a proportional or non-proportional font. Ideally it would be a non-proportional font since these tend to look better in a non-proportional display. The more important point though it is that you choose a font which contains all the characters you require. For a discussion on this topic please see “Displaying non-native codepages“ on page 22.

18.4.2. Set Colours

Displays a dialog allowing the setting of the colour of various parts of the editor. Please see “Colour Dialog“ on page 134 for more information.

18.4.3. Queues Prefix

By default **MQEdit** will attempt to retrieve all queue definitions. However, there can be occasions whether it can be preferable to only query a subset of the queue names. This can be because of security considerations since querying all queues might generate authority events on the Queue Manager. Or it could be for efficiency reasons where the Queue Manager has an enormous number of queues and you are only interested in a small portion. By entering a prefix value in this field and requesting a queue refresh MQEdit will just request a refresh of the queues starting with the given prefix.

18.4.4. Search Locations

When this option is selected a search field is shown at the top of the main window Queue Manager navigation pane. Entering text in the search field will automatically cause the list to be refreshed so that only locations containing the text will be displayed. The search text can contain the standard wildcard characters '*' and '?'. The full format of the search string is a comma separated list of search strings followed by an optional search qualifier which says which location field should be compared. Without a qualifier MQEdit will check all the fields below.

Location Field	Qualifier	Qualifier Shortform
Queue Manager name	qm	m
Location	loc	l
Group Name	grp	g
Network Names	net	n
Client Connection Name	conn	c

If you choose to use a qualifier it should be specified between < > characters.

Examples of search strings might be:

Search String	Meaning
abc	Search all fields for the text 'abc'
abc <qm>	Search only the Queue Manager name for the text 'abc'
abc,def	Search all field for either text 'abc' or 'def'
abc,def <qm loc>	Search for either text 'abc' or 'def' but only look in the Queue Manager or Location fields
abc,def <m l>	As above but using short form qualifiers
abc*def	Search for a single string containing 'abc' and 'def' with zero or more characters between them
abc?def	Search for a single string containing 'abc' and 'def' with a single character between them

Searches are case insensitive. The search will applied after a short delay.

If you wish you can save a default search value by setting the required search field and then selecting *View-Save search values* from the main menu.

18.4.5. Search Queues

When this option is selected a search field is shown at the top of the main window Queue Manager navigation pane. Entering text in this search field will cause only queues which match the text to be displayed. The search text can be a comma separated list of strings containing the standard wildcard characters '*' and '?'.

For example, you could search for **SALES,TEST** which would only show queues which had either the string SALES or TEST in their name.

If you wish you can save a default search value by setting the required search field and then selecting *View-Save search values* from the main menu.

18.4.6. Save search values

Selecting this menu with save the current search values as the default values. The default values are the values that will be used when MQEdit is first started. In addition the search fields can be set back to the default values at any time by selecting *View-Restore default values* from the menu.

18.4.7. Restore search values

If this menu is selected then the search fields will be replaced with the current default search values and the Queue Manager panel will be changed to reflect the new values.

18.4.7.1. Show Queue Depth

When this option is active the current queue depth of the queue is added to end of the displayed entry. Note that this is just a snap-shot of the current depth and is not automatically updated. Press the refresh button on the required Queue Manager to get the updated depths.

18.4.8. Location Display

These menus control how the location line in the Queue Manager navigation pane looks. The settings are remembered across re-starts of the editor.

18.4.8.1. Include Queue Manager Name

When selected displays the Queue Manager name

18.4.8.2. Include Location

When selected the editor displays the 'Location' description against each Queue Manager.

18.4.8.3. Include MQ Version

When this option is active the MQ Version (if known) of the location is added to end of the displayed entry.

18.4.8.4. Include Queue Count

This option controls whether MQEdit will display the number of queues available next to each Queue Manager entry. Note that this number will probably not be the actual number of queues on the Queue Manager. For example, Dynamic queues are usually not displayed in MQEdit. In addition security settings and filtering can affect how many queues are available for use by MQEdit.

If 'search queue' is active then the count is displayed in two parts, eg. (12/173). The second number is the total number of queues available as before. The first number is the number of queues which match the search field and are therefore actually displayed in the panel.

18.5. Windows

Menu items concerned with window and dialog management.

18.5.1. Close all

Closes all windows

18.5.2. Minimize all

Minimizes all windows

18.5.3. Tile Horizontal all

Tiles all the non-minimized windows and favours making the windows 'horizontal'. The window include the main windows so if you have multiple instances of the main window this can be a useful way of organising the display.

If windows are not put in the ideal position then two windows can be 'swapped' by moving the window with the <Ctrl> key held down. The window swapped with the window whose top left is closest to the top left of the window being moved.

18.5.4. Tile Vertical all

Tiles all the non-minimized windows as above but favours making the windows 'vertical'.

18.5.5. <Windows>

Most dialogs and windows opened in MQEdit will be displayed in this list. By selecting an entry the window will be brought to the foreground and restored.

Chapter 19. Preferences Dialog

The preferences dialog allows the user to set the behaviour of various aspects of the application. The dialog is split into a number of different parts. The first being a set of on/off switches in a scrollable list. The item is on if a tick is displayed next to the item.

19.1. General

19.1.1. Auto save configuration

When checked MQEdit will automatically save the configuration when a significant change, for example adding a new location, is made.

19.1.2. Import local Queue Managers on start-up

This option controls whether MQEdit will check whether there are any local Queue Managers defined which do not exist in the MQEdit configuration. If there are missing Queue Managers a dialog will be displayed on start-up allowing the user to import the definitions.

19.1.3. Show Permanent Dynamic Queues

This option controls whether permanent dynamic queues should be displayed in the Queue Manager panel. Generally speaking Permanent Dynamic Queues, like Temporary Dynamic Queues, are used solely for application reply queues. As such they rarely contain messages of interest. In addition, in some systems, you can get a very significant number of these dynamic queues. To aid with clarity, and performance, these dynamic queues are therefore not shown by default.

19.1.4. Use INQUIRE QUEUE NAMES

By default MQEdit will issue an INQUIRE QUEUE command to retrieve the queue names for a Queue Manager. In most situations this is fine but it does have a couple of disadvantages. Firstly, IBM MQ will raise an authority event for any queue which matches the command but that the user does not have the authority to view[21]. If your installation is well managed and the authorities of your queues are tightly controlled this can mean that a single INQUIRE QUEUE command can generate hundreds or even thousands of authority events! What many installations do is just catch these events and immediately discard them if they are a DISPLAY command. However, this has the potential to obfuscate the real hack attempt of someone displaying objects. The second disadvantage of INQUIRE QUEUE is that each queue name is sent as a separate MQ message. For large installations this can mean that many thousands of messages are generated just for one command.

Telling MQEdit to use INQUIRE QUEUE NAMES solves both of the issues. INQUIRE QUEUE NAMES will not generate authority events and the response is a single message so it is much quicker and more efficient. However, since this is real life, there are of course downsides. The principal one being that MQEdit is told only the queue name and queue type. It is not told the current queue depth nor the def type of the queue. This means that if you use INQUIRE QUEUE NAMES MQEdit will not be able to show a queue depth next to the queue definition nor will it be able to filter out temporary dynamic queue definitions.

19.1.5. XML Assist

When checked, MQEdit will do the following actions automatically when you are editing XML in a message.

- Insert XML tag markers when you type the first ',<' character.
- Update both tags when you edit one of them.

19.1.6. Show offsets in hex

By default MQEdit will display message offsets in hexadecimal. However, if you prefer you can uncheck this options and MQEdit will display the offset in decimal.

19.1.7. Wide insert cursor

By default MQEdit will display an insert cursor which is the width of the border of the current display scheme. This can be as little as one pixel wide in some displays which may make the cursor hard to see. Selecting this option will always display a two pixel wide cursor.

---

[21] You would not be alone if you thought this was entirely unnecessary and, one might say, misleading. If a user doesn't have authority to view queue X and issues a DISPLAY QUEUE(X) command then an authority event is entirely appropriate. However, if the user issues the command DISPLAY QUEUE(*) then raising an authority event because the user doesn't have authority to view a queue that would have been returned seems overkill. The user is clearly not trying to 'hack' the definitions in any sense they just want to view what queues are available to them. This issue has been raised with IBM MQ Development and they did add the ability to prevent any authority events from a DISPLAY command. Sadly it will prevent DISPLAY QUEUE(X) just as well as DISPLAY QUEUE(*) but this may be OK for many people. If you want to read further about this subject then we have written a blog post about it here.

19.1.8. Remove context override on each message
By default MQEdit will re-apply the message context Preference settings (see below) each time the user edits a different message. This can prevent inadvertently using the wrong context setting when moving to a new messages. However, the user might prefer that the settings made in the edit dialog persist during the whole editing session. If that is the case then uncheck this option and the preference settings will only be adopted for the first message.

19.1.9. Initial Add Context
This value controls what should be the initial value of the message context mechanism used when creating a new message.

The default is “Default” which means that IBM MQ will supply the normal application context fields.

19.1.10. Initial Update Context
This value controls what should be the initial value of the message context mechanism used when editing an existing message.

The default is “Pass All” which means the context fields are left as the values of the original message.

19.1.11. End Of Structure Detail Level
This setting allows you to choose whether MQEdit displays a line after each structure giving it's length in bytes. By default this line is only shown at the highest detail level however you could choose to display it even at the lowest detail level, or perhaps not at all.

19.1.12. Max Hex Dump Bytes
By default MQEdit will display as many bytes on a line as will fit the width of the screen. This is to make most use of the available screen real estate. However, if you prefer, you can set this value to maximum number of bytes you would like MQEdit to show on each hex dump line. This can be useful if, for example, you would like MQEdit to match your normal hex dump format.

19.1.13. Notify of program updates
MQEdit has the ability to periodically check for program updates. This allows you to set three different modes.

➢ **Never**	MQEdit will not check for updates
➢ **New version only**	MQEdit will only notify the user if a new version is available
➢ **New version or build date**	MQEdit will notify the user of any newer version

Clearly this feature requires an available and working internet connection. If, for some reason, the web page can not be accessed then a message box explaining the issue will be displayed to the user.

19.1.14. Startup Splash, Qm Normal Splash, Qm Specific Splash
These settings are all concerned with displaying a user configurable splash screen to the user. For more information about splash screens please see “Splash Screens“ on page 130.

19.1.15. Licence Command
This setting allows the user to specify a command which will run should there be either an invalid licence or it is soon to expire. For more information please see “Licence Command“ on page 132.

The licence command is followed by a 'Test' button. This button has two purposes. Firstly it allows you to run any configured licence command and will display an error dialog if there are any problems. Secondly, whether you have configured a licence command or not, it will re-load any configured licences. This allows you to change the install licences without having to end the program.

19.2. Confirmations

19.2.1. General
Confirmations associated with the program in general.

19.2.1.1. Confirm on program exit
When checked MQEdit will always ask for confirmation before ending. This can be useful to prevent accidental ending of the program.

19.2.1.2. Show confirmation dialogs
When checked the application will display a confirmation dialog whenever a potentially harmful command such as “delete message” is issued. You can disable individual confirmation dialogs below.

19.2.2. Message Operations

Confirmations associated with message operations such as move, copy and delete.

19.2.2.1. Confirm message copy

When checked MQEdit will confirm before it copies messages from one queue to another.

19.2.2.2. Confirm message move

When checked MQEdit will confirm before it moves messages from one queue to another.

19.2.2.3. Confirm message delete

When checked MQEdit will confirm before it deletes any messages.

19.2.2.4. Confirm message 'copy all'

When checked MQEdit will confirm before it copies all the messages from one queue to another.

19.2.2.5. Confirm message 'move all'

When checked MQEdit will confirm before it moves all the messages from one queue to another.

19.2.2.6. Confirm message 'delete all'

When checked MQEdit will confirm before it deletes all the messages from a queue.

19.2.3. Message Editing

Confirmations associated with editing a message.

19.2.3.1. Confirm message add

When checked MQEdit will confirm before it adds a new message to a queue.

19.2.3.2. Confirm message update

When checked MQEdit will confirm before it updates the message on a queue.

19.2.3.3. Confirm unapplied edit

When checked MQEdit will confirm that you want to continue if you have made some edit changes that have not been written to a queue.

19.2.3.4. Confirm retrieval of full message

If you start editing a truncated message then MQEdit will automatically retrieve the full message. This setting controls whether MQEdit asks for confirmation before it does so.

19.2.3.5. Confirm add of truncated message

When checked MQEdit will confirm that you want to continue if you try to add a truncated message to a queue.

19.2.3.6. Confirm edit of converted message

Generally speaking it is not recommended that you edit a converted message since the actual conversion process may have changed the message and you are not therefore seeing the original message. However this could, of course, be exactly what you want to do. For example, you want to convert a message to send it to another queue because the application there does not use MQGMO_CONVERT. However, since conversion is generally not advised then this option will warn you if you start editing a message which have been converted.

19.2.3.7. Confirm edit of message in character substitution mode

Substitution mode is a mode where sequences such as &gt; are displayed as >. Editing in this mode can be hazardous since what is displayed to the user is not necessarily exactly what is in the message. For more information please see 'Character Substitution Mode' on page 37.

19.2.3.8. Confirm put of a message with formatting error

MQEdit will detect certain types of errors when formatting the message. For example in an XML message end tags could be missing or in a PCF message you could have invalid structure lengths. When this confirmation is checked MQEdit will ensure that you don't accidentally put a message with one of these formatting errors. Note that a message which is formatted in hex will never have an error since it is just a series of bytes. If you want MQEdit to check for correct formatting then you should put the message while the editor is showing the formatted display.

19.3. Dialogs

19.3.1. Save Dialog Positions

When checked the positions of the command dialogs will be saved

19.3.2. Save Dialog Sizes

When checked the sizes of command dialogs will be saved

19.3.3. Integral monitor positioning

Normally MQEdit will try to ensure that new dialogs are initially presented so that they do not span between multiple monitors, even if that was the last saved position of the dialog. By unchecking this option this behaviour is switched off and dialogs will be presented exactly where they were last.

19.3.4. Adjust Dialog positions for dual monitors

When this option is checked MQEdit remembers the position and size of dialogs independently for when the programs is running with one or two monitors. This reduces the chance that a new dialog is placed on the second monitor which isn't there any more.[22]

19.3.5. Show object name field as a drop-down list

When checked the main object name, such as queue name, in a dialog is shown in a dropdown list. This has the advantage that the list of recently used names can be scrolled through. However, it does have the disadvantage that the value can be inadvertently changed by moving the scrollbar. If required this option can be unchecked which will display the field in a normal entry field.

19.3.6. Show thousands separators

When checked numerical values in dialogs will contain thousand separators. So, for example, by default 4194304 will be displayed as 4,194,304. If you wish you can change the separator character which is used at the bottom of this tab.

19.3.7. Display dialogs on taskbar

By default the dialog windows are displayed on the windows taskbar. However, if you have many of them then this can use up valuable taskbar space. By deselecting this option the dialogs will not be on the taskbar but can be easily accessed via the 'Windows' menu item on the application main window.

19.3.8. Ticks should be green

When selected, tick symbols ✓ (sometimes known as check marks) are displayed as green. When not selected the symbol will be displayed as red. Remember that until you press 'OK' or 'Apply' no change will take place.

19.3.9. Show refresh time in titlebar

This option controls whether MQEdit will display the last refresh time in the titlebar.

19.3.10. Refresh time on the right

By default MQEdit will try to paint the refresh time to the right of the titlebar. However, this facility may not be available in some Windows themes. In particular Window 7 Aero themes don't allow applications to draw over the non-client area of the titlebar. To see the refresh time you may need to switch to the 'classic' theme or uncheck this option so the refresh time is part of the actual dialog title.

19.3.11. Show tooltips as balloons

When a list dialog contains more columns than can be displayed across the screen the titles of the columns are compressed. Hovering the mouse over the column title will display the full name of the field in a tooltip. This option controls whether the tooltip is displayed as a balloon or a simple rectangle. Note that changing this option will only take effect the next time the application is started.

19.3.12. Up/Down keys should tab between fields

This option controls whether the up/down keys to tab between fields in a command dialog. The disadvantage of this is that to select from a pull-down list you actually need to bring the pull-down list down. By disabling this option the user can scroll through the available choices of a pull-down with the Up/Down buttons. The tab button can always be used to tab between fields.

---

[22] Note that MQEdit does not actually know whether a second monitor is physically connected or not. If the Windows settings says there are two monitors but there is only one connected then dialogs will still be placed on this non-displayed monitor.

19.3.13. <esc> should end dialog

When selected the escape key will end the current dialog.

19.3.14. Allow selection of read only fields

This option changes whether read only fields in the object dialogs can be selected. Allowing them to be selected means there can be more fields to tab over but it also means that you can select text to copy/paste into other non-read only fields.

19.3.15. Show existing dialog by default

When the user requests a dialog, for example a Queue Manager dialog, this option controls whether any existing Queue Manager dialog for this location should be displayed or whether a new one should be created. Holding down the <Shift> key when the dialog menu is selected will give the alternate behaviour than specified by this option.

19.3.16. Disable read ahead

Read Ahead can give significant performance advantages particularly over slow networks. However, it can also cause more data to be retrieved from the server than necessary. When checked this option switches off read ahead for all locations, both the reply queue processing and browsing message queues. For more granular control the user can also disable read ahead by location in the option section of the location dialog. For more information please see "Location Dialog" on page 116.

19.3.17. History of Object Names

How many object names should be kept in the drop-down history

19.3.18. Progress Dialog Delay

This is the number of seconds a message operation should should run before the progress dialog is shown. The default value is 3 seconds. A value of 0 will display the dialog immediately. A value of 'Never' requests that Progress Dialogs are never created. Setting a large value, for example, 99999, will effectively only display the dialog if the user explicitly chooses it from the 'Windows' menu of the main window.

19.3.19. Separators

19.3.19.1. Thousands

Here you can specify the character which is displayed as the thousands separator if thousands separator option is checked. This allows you display numbers such as 4194304 as 4,194,304. You can specify up to 3 characters. The default is ','.

19.3.19.2. Lists

Here you can specify the characters which are displayed between a list of values. You can specify up to 3 characters. The default is '; '.

19.3.19.3. Indicators

Here you can specify the characters which are displayed between indicator values. You can specify up to 3 characters. The default is ' : '.

19.4. Lists

19.4.1. Right justify numeric columns

When selected the majority of numeric data will be shown right justified. Exceptions to this are 'identifier' type numerics such as monitor identifier, queue position and codepage.

19.4.2. Lists should be striped

When checked list dialogs are candy striped. The main colour and the stripe colour can be set in the View->Set Colours… menu item. Please see "Colour Dialog" on page 134 for more information.

19.4.3. Show rule lines on lists

When checked list dialogs will be displayed with a ruled line underneath each item displayed in the list. The colour of this line will be the list selection background colour.

19.4.4. 3D List Titles

When checked the titles in list dialogs appear like three dimensional buttons. Regardless of this setting the titles will always behave like buttons and allow sorting based on the field title selected.

19.4.5. 3D List Sort Arrow

For most colour schemes the arrow displayed in the list titles to identify the sorted field looks better when displayed in a 3D fashion. However, for some colour schemes it may be preferable to switch off the 3D attribute.

19.4.6. Filled List Sort Arrow

For the same reason as the previous option it may be preferable to colour fill the sort arrow with the foreground colour.

19.4.7. Select first entry on refresh

If checked the first entry of the lists will be selected when the list is refreshed if the list has no current selection.

19.4.8. Raised, Bump, Sunken, Etched, Flat, One dimensional, Half, Softer

These options can be used to change the way the list items are displayed. These options control calls to the windows DrawEdge function and the exact display results depending on the implementation of that system. In particular, in my experiments, **Softer**, has no effect on most of the border types.
The **Half** option causes the border to be drawn only on the bottom and right hand sides of the box. This means that although the appearance is softer it also reduces the 3D look of the border.

19.4.9. 3D Selection

When selected the selected items in the dialog listbox will be drawn in a 3D fashion. The effectiveness of this option will depend on the colour scheme being used.

19.4.10. Reposition list on sort

If checked the list is repositioned, after a sort, to try and display the same entry as displayed before the sort. If not checked the top of the list is shown after sorting.

19.4.11. Size columns by data portion only

When displaying a list of objects the column widths are automatically set by the greater of the title width and data width. Some fields, generally numerics, tend to be quite narrow despite having fairly large titles. By selecting this option the column width will only be sized based on the data rather than the title. Note that this can make the title unreadable.

19.4.12. Use Global filter history

If checked then the history of filters used in the list dialogs will the same for each location. If unchecked then a separate list will be maintained per location. Holding the 'Alt' key when showing the filter history will show the 'other' history. So, for example, if you chose to use the non-global filter history then by holding the 'Alt' key as you displayed the filter list the displayed list would be the global list.

19.4.13. Import shared filters

This option controls whether MQEdit is a generator or a consumer of shared filters.
For a description of shared filters please see Shared Filters on page 69.

19.4.14. History of filter expression

How many filter expressions should be kept in the drop-down history

19.4.15. List Flash Interval

How many seconds between flashes for for lists that have used the flash() or flashcell() functions.

19.4.16. Shared Filter File

The location of the shared filters. For a description of shared filters please see Shared Filters on page 69.

19.5. Browse

19.5.1. Convert EBCDIC for display

If checked then MQEdit will try to display the EBCDIC data in messages as readable characters. This will probably make messages easier to understand but can be misleading as to their content.

19.5.2. Warn about possible data conversions

If selected then MQEdit will write a warning message in the status display if some of the characters shown have been converted from EBCDIC by MQEdit. In addition, the text '(after conversion)' is added to any CCSID values if conversion is selected and the code page is the local codepage. This text reminds the user that the message on the queue may be in a different code page.

19.5.3. Auto update browse lists

When checked the application will refresh browse queue displays if actions may have caused the contents to change. For example, putting, copying, moving or deleting messages would update the queue browse display. The application can not always determine which lists may have changed however so the user should periodically refresh the display or set auto-refresh on the dialog if you want to be certain you are looking at the latest data.
Auto updates can be disabled per location in the location dialog.

19.5.4. Default Msg Display Range

This option allows the user to set the default browse range when browsing a queue. By default it has the value ‘1,100’ which is the first 100 messages. Any value can be set but bear in mind that large values could cause large numbers of messages to be retrieved from the server with the related impact on performance.

19.5.5. Default Max browse message size

When browsing messages in a queue the message size is limited to a maximum size which is displayed on the dialog. This prevents accidentally downloading very large messages from the queue and enables quick navigation of the messages. The default maximum message size is 10,000 bytes but if you frequently want to browse messages larger than this you might like to increase this default size.

19.5.6. Default Max Browse Messages

MQEdit supports browsing up to 30,000 messages at once. However, the default is 10,000 to avoid unnecessary delays when browsing deep queues. Essentially this value controls how wide the message display range can be. Even with a small maximum browse range, and therefore a small display range, it is still possible to browse deep queues. For example, to see a few messages near the 100,000th message one could specify a range of 99995,100005.

19.5.7. Message Summary Size

Message summary is feature which summarizes the message based on it's first few bytes. For example, a message on the Dead Letter Queue will have a header which identifies the cause of the message being put on the Dead Letter Queue. A message on the transmission queue has a header which identifies where the message it going. This setting controls, in a message list, how many bytes of the message are requested so that the message can be summarized. Of course the higher the setting the more detailed the message summary for some messages at the cost of retrieving more data from the server.

The minimum value is 108, the maximum value if 500 and the default value is 120.

19.5.8. Message Summary Display

This setting controls how many bytes of message data is displayed if the message can not be summarized in the browse message list. If a message can not be summarized, for example it is just a bytes message with no format or is not a recognised format, then MQEdit will just display the first few bytes of the message as characters (if they are printable). This setting allows you to set how many of those bytes are displayed.

19.5.9. Locale

Any valid Windows locale string can be entered in this field. The value specified will control the characters displayed in dialogs such as queue browsing. It will also control which characters are considered as printable. A value of ‘Default’ returns the application to the default windows setting.
The drop down list contains the basic settings for this field. Selecting one and pressing apply will then display the locale selected by windows. The number at the end of the locale is the codepage. This can be changed if required provided the entire string is recognised by windows as a valid locale string.

19.5.10. Browse CCSID

This field allows you to specify a codepage to convert messages to when browsing messages on queues. In general if you change the value of ‘locale’ you would want to change this CCSID to match the codepage number. A value of ‘Default’ will cause the application to use the standard Windows codepage setting.

19.6. Connection

19.6.1. Auto start command server

With this option checked the application will check whether the command server is running when it connects to a local Queue Manager**.** If it isn't it will be started with the 'strmqcsv' command.

19.6.2. Uppercase MODEL Queue userid

When using a model queue as a location reply queue the application will construct the name of the queue based on the application name and the userid running the program. For some security profiles it may be useful to always have the userid folded to uppercase.

19.6.3. Use MQSET to free reply queue

When MQEdit tries to open a reply queue for a location it might find that queue already in use. This may happen because a previous instance of MQEdit was not ended cleanly. By default, MQEdit will then inhibit get on the reply queue forcing any outstanding get to return. This then allows the current instance of MQEdit to open the reply queue successfully. The ability to issue MQSET against a queue does, however, require a higher level of authority which may be inconvenient. This MQSET processing may be switched off by unchecking this option.

19.6.4. Reconnect on 'not authorised' failure

This option controls whether MQEdit should reattempt the connection if it gets a 2035 (not authorised) failure.

19.6.5. Auto Connect Delay

This parameter sets the frequency, in seconds, an auto connect location will try to connect to its target Queue Manager. Please see "Time Interval" on page 137 for the format.

19.6.6. Command Expire Interval

This parameter sets the expiry interval of the command messages sent by the MQEdit program. Expiring messages is useful to prevent a build-up of messages on, for example, a transmission queue which can't connect to a remote location.

19.6.7. Password Cache File

These fields control the use of the password cache file. For more information see Chapter 22. Password Cache File on page 123.

19.7. Export

19.7.1. Export Text Title

Controls whether a title line is written preceding any exported data when exported in Text format.

19.7.2. Export MQSC Title

Controls whether a title line is written preceding any exported data when exported in MQSC format.

19.7.3. Text Title

This field allows you to set the format of the title line exported with any text exports. The field can contain inserts please see "Text Inserts" on page 135 for a description of these inserts.

19.7.4. Text Sample

This field shows an example of how the title line or lines would look.

19.7.5. Date Format

This field selects the format which should be used in the date column of the auto-export timestamp.

19.7.6. Date Separator

This field selects what separator character should be used in the date column of the auto-export timestamp.

19.7.7. Date Example

This field shows what the auto-export timestamp date would look like.

19.7.8. CSV Separator

The character to use for comma separated lists generated by the export command.

19.8. User Formats

This tab contains values related to define user message formats to MQEdit.

19.8.1. Allow user formats

This setting controls whether user formats are applied or not.

19.8.2. Autodetect EDIFACT messages

This setting controls whether MQEdit should interrogate messages for the telltale start of an EDIFACT message.

19.8.3. Display user format summary fields

When you define a user format you can identify which fields, if any, should be displayed in the message list pane view. This option controls whether these summary fields are honoured or whether the message is just displayed as text.

19.8.4. File Name

This is the file that MQEdit should read for the User Message Format definitions. An error will not be generated if the file is not found. For a description of the file format please see 'User Format Messages' on page 72.

19.8.5. Reload User Formats

Pressing this button will cause MQEdit to reload the User Formats definitions from the definition file and reapply the definitions to any current messages. This button can be very useful when developing the file definitions to see whether the output of MQEdit is as required.

19.9. Sounds

19.9.1. Warning Sound

The sound played when the application wishes to issue a warning. For example an invalid value has been entered or a command failed.
The value can be beep, none, or .WAV file.
Depending on the OS it may be necessary to enter the full path to the .WAV file
The application will play the sound when the user double-clicks on the entry field or presses the 'Play' button.

19.9.2. Alarm Sound

The sound played for the application alarm.
The value can be as above for the warning sound.

19.9.3. Sound Times

Allows the specification of how long and how often the sounds should be generated.
The format of the field is **X [: Y [: Z]]** where :-

- **X** **Minimum alarm interval**
  Minimum time which must have elapsed before sounding the alarm again
- **Y** **Maximum alarm period**
  Maximum time to sound the alarm for
- **Z** **Minimum sound interval**
  Minimum time which must have elapsed between making any sound

19.10. Display

19.10.1. Tabbed

If checked then some of the dialogs will be displayed using TABs to group together relate fields in categories. These dialogs have large numbers of fields and using TABs can make finding a particular field much easier. If you uncheck this setting then dialogs are shown as a just a list of fields.

19.10.2. Multi-line

If TABs are switched on then this setting controls whether the TABs should be displayed across multiple lines or as a scrollable list.

19.10.3. Bottom

For horizontal TABs this setting controls whether the TAB is draw at the top or at the bottom.

19.10.4. Buttons

This setting controls whether the TABS are actually shown as buttons.

19.10.5. Flat

If the TABs are displayed as buttons this setting sets whether they should be displayed as normal buttons or as flat buttons.

19.10.6. Vertical

This setting controls whether the TABs are drawn horizontally or vertically.

19.10.7. Right

For vertical buttons this setting controls whether the TABs are drawn to the left or the right.

19.10.8. System Draw

TAB controls in Windows are quite odd little controls and are, quite frankly, riddled with problems. One of the main problematic areas is changing their appearance. A simple search on the internet will find dozens of confused programmers tearing their hair out wondering why the control is not behaving. In the end, some of them do as I have done and take on the responsibility of drawing the TAB themselves. So, by default, MQEdit draws all the TABs in the dialogs itself. That way the TAB can be drawn 'correctly' and using the configured colours. This has the advantage that the TAB fits in with the colour scheme. The disadvantage of this approach though is that the TABs may not look like other TABs in the Windows system. So, this setting allows the user to choose whether they prefer the command dialog TABs drawn by Windows itself which don't conform the the colour scheme or TABs drawn by MQEdit which may not look exactly like other TABs on the system.

19.10.9. Toolbar Raised, Highlight, Highlight Border
Selects the style that tools are displayed on the toolbar.

19.10.10. Below Edge
Determines the type of edge which should be used on the bottom of the toolbar. Note that you can deselect all.

19.10.11. Above Edge
Determines the type of edge which should be used on the top of the toolbar. Note that you can deselect all.

19.11. Time

19.11.1. Large Format, Medium Format, Small Format
These values control what is displayed on the right hand side of the dialog titlebar to display the last refresh time for the message list[23]. Any character string may be entered with the addition of certain inserts preceded by a '%' character. The inserts characters are :-

Insert	Meaning
H	Two digit hour
M	Two digit minutes
S	Two digit seconds
d	Two digit day of month
m	Three character month name e.g. Jan, Feb, Mar….
y	Four digit year
D	Three character day of week e.g. Mon, Tue, Wed…
t	Simple time format e.g. 18:14:03
%	Percent character

A preference option controls whether the refresh time is shown on the left or the right of the titlebar. If the time is shown on the left then only the small time format is used.

19.11.2. Time Zone
Only available if the user is showing local times and not using the system setting for local time-zone. Enter the number of hours (positive or negative) from GMT using a real number. For example, a value of -1.5 would represent 1 hour, 30 minutes before GMT.

19.11.3. Show local times
When checked the values for GMT times, such as put time in a message browse will be displayed in local time rather than GMT.

19.11.4. Use System settings
When checked the value for the local time-zone will be taken from the windows system. This value must be unchecked to be able to specify your own time zone.

19.12. Help

19.12.1. Help file command
This field is provided when help is provided by the PDF file. If can be left blank if you machine has an association that knows what program to run for PDF files. If no association exists then enter the program that should be used to read the PDF file.

19.12.2. Help file location
Enter the location of this manual PDF file. The file is displayed when the user presses 'F1'.

---

[23] This will not be displayed if the desktop is configured to use a Windows Aero 'glass' theme. This is because the Windows OS prevents the application painting to the non-client area of a window.

Chapter 20. Location Dialog

The location dialog is where you can add and change the definitions of the Queue Managers you wish to be able to contact. A description of each fields is given below.

20.1. Fields

20.1.1. Location

A free format text description of the location. This field is to allow the user to assign a more meaningful name than just the Queue Manager name.

20.1.2. Queue Manager

The Queue Manager name that should be used by MQEdit when it tries to connect to the Queue Manager.

20.2. Connection Tab

20.2.1. QM Group

An optional free format group name for this location. The effect of this value is that the location will be displayed in Queue Manager navigation tree view ‘under’ the group name. This allows all your z/OS machines, for example, to be contained in a group such as “z/OS Queue Managers”. Groups are always placed at the top of the Queue Manager navigation pane.

20.2.2. Network Names

You can give a comma or space separated list of Network names with which you want this location to be associated with. This allows groups of Queue Managers to be associated with particular network name.

20.2.3. Reply Queue

This is the name of the predefined reply queue on the above Queue Manager. It defaults to the name SYSTEM.DEFAULT.MODEL.QUEUE. See “Reply Queue Details” on page 128 to see how another name can be used.

20.2.4. Reply Prefix

If a model queue is given in the Reply Queue field then the Reply Prefix field can be used to specify the prefix which should be use to construct the actual temporary queue name. See “Model Reply Queue” on page 128 to see how another name can be used.

20.2.5. Command Queue

The name of the queue at the location which processes the MQ commands. This value is set automatically to the appropriate name depending on whether the MVS check box is selected or not. It can be changed to another queue if required.

20.2.6. Client

If checked this indicates that MQEdit should connect to this location via a client connection rather than directly.

20.2.7. Configure(d)

Only available if ‘Client’ is checked
If the location is to be connected to via a client then the client definition to be used can be entered in a dialog presented when this button is pressed. If a separate client channel definition is not entered then the client connection will use the normal MQSERVER or Client Channel mechanisms to connect to the server.

If the target location is a multi-instance Queue Manager then the connection list field should be a comma separated list of the IP addresses of the locations where the Queue Manager can be located.

The name of this button will either be ‘Configure’ or ‘Configured’ depending on whether there is currently a client configuration specified.

20.2.8. CCDT URL

This field can be used to specify the location of the CCDT file to use. The field is only enabled when the 'Client' check box is checked and there is no local client definition specified.

20.2.9. MVS

Select this if the remote location is z/OS. This will cause the name of the command queue to change. When checked, messages will be sent to the command queue in MQSC, rather than PCF format.

20.2.10. Auto Connect
When checked the application will always try to ensure that it is connected to the Queue Manager. If the button is unchecked then connection to the Queue Manager will be made ‘when necessary’. In other words the connection attempt will only be made when a queue is browsed or a command is issued against this Queue Manager.

20.2.11. Retry Interval
If the connection to the queue manager fails for some reason (for example, the Queue Manager is ended) then this field allows the specification of a number of seconds before a reconnect attempt is made. No automatic reconnection attempt will be made if the value is 0.

20.2.12. Disconnect Interval
This field allows the specification of the number of seconds of inactivity before MQEdit disconnects from this location. A value of 0 means that the editor will never disconnect.

20.3. Security Tab
The security tab of the location dialog contains fields which are primarily concerned with the security of the connection. However, these are by no means the only fields associated with a location which are related to security. For example, the channel definition contains a number of fields concerned with security such as the SSL/TLS values which affect encryption levels and authentication across the client link.

20.3.1. Userid and Password
Check this item if you wish the program to pass a userid/password combination over on the connection. This option requires at least MQ Version 6 to be installed on both the client and the server. The program will remember the userid across invocations of the program. The password will only be remembered across restarts if a password cache file is configured.

The userid and password is always cached for the life of the program so that the user is only prompted to enter the details as infrequently as possible.

20.3.2. Upper Case Userid / Password
Check one or both of these items if you wish the Userid and/or Password fields to be automatically folded to upper case before being sent to the queue manager. This can be especially useful for z/OS TSO logins where it can be easy to forget that you have an upper case password because the TSO login panel always folds it to upper case for you.

20.3.3. Security Exit Only
This field only applies if the 'Userid' checkbox is checked. This option controls whether the userid is passed solely to the security exit or whether it is also passed to the Object Authority Manager (OAM) component of the Queue Manager. Essentially this option controls whether the MQCSP_AUTH_USER_ID_AND_PWD option is specified when MQEdit connects to the Queue Manager.

20.3.4. Cache on Success Only
This field only applies if the 'UserId' checkbox is checked. This is to avoid the user having to specify the values multiple times, each time MQEdit makes a connection. This option controls how MQEdit caches any entered Userid/Password. If selected then the userid will only be cached if the connection is successful. If unchecked then the userid will be cached unless the connection attempt fails with either MQRC_NOT_AUTHORIZED or MQRC_CONNECTION_NOT_AUTHORIZED. If you are using a Channel Security exit to validate your userid/password then you probably want to check this option because the exit can't cause these reason codes to be returned.

20.3.5. Store in cache file
If checked then MQEdit will retrieve and store any entered userid and password for this location to the defined password cache file if it is defined. The password cache file is set in the Connection tab of the Preferences Dialog.

20.3.6. Security Group
The idea of a Security Group is that a number of Queue Managers may be secured using the same mechanism and therefore it useful if they can share the same userid/password.

- **Name**
  All locations with the same Security Group Name will be considered part of the same group. The actual name is entirely arbitrary and need not have any bearing on other security values/domains. It is just a name which MQEdit will use to group

locations together. Changing the userid/password for one member of the group will change it for all members. If the password cache is being used then just one entry for the entire group is made.

- **Description**
  Having a description for your Security Group in entirely optional. If specified then the user will be presented with this description when asked for a userid/password so it can be useful as a memory jogger.

20.3.7. Set Password

Pressing this button will allow the user to enter a new userid/password combination for this location. The values will be cached if configured to do so.

20.3.8. Delete Password

Pressing this button will remove any record of the userid and password from the cache.

20.3.9. Allow Weak Cipher Specs

A number of Cipher Specs (those using the SSL V3.0 protocol, and those using no encryption, or weak encryption algorithms) are no longer recommended since they don't really provide sufficient protection for sensitive data. As a consequence by default they will not be presented as choices in new channel definitions. However, if you have a need to define channels using these old Cipher Specs then you can select this option. Note that, depending on the version of MQ you are defining the channel on, you may need to also configure MQ to accept these weak Cipher Specs. For further information please see https://mqgem.wordpress.com/2015/07/21/deprecated-cipherspecs/

20.3.10. Forget KDB Password

If you have provided MQEdit with the password for a Key Database file which has subsequently changed, you can cause MQEdit to forget any current knowledge of that KDB file and prompt you for a new password next time it is needed, by pressing this button. The same KDB file (and thus password) is most likely being used for many queue manager connections in MQEdit. Pressing this button forgets the password for all those queue manager connections as it is stored only once for all to use.

20.4. Options Tab

20.4.1. Disable Auto Updates

Auto updating lists can be a useful convenience. However, refreshing lists on some locations can be expensive, for example a client connection over a slow link. In these cases this option can be used to switch it off just for a particular location.

20.4.2. Download Objects

When selected the key fields of the queue definitions will be downloaded soon after connection is made with the Queue Manager.

20.4.3. Disable Read Ahead

When selected this option will disable read ahead when browsing queues from this location. The option only has an effect when the location is a client connection. Read ahead can give a significant speed improvement when browsing queues, particularly over a slow network. However, it can mean that more messages than necessary are sent to the client from the server. For this reason the user is able to disable read ahead processing.
An option in the preference dialog allows the user to disable read ahead for all locations, please see “Preferences Dialog“ on page 105 for more information.

20.4.4. Upper Case

These options allow the user to say whether, by default, characters should be entered into the object names or exit names. in upper case. When selected pressing the *‘shift’* key will allow entry of lower case characters.

20.4.5. New Line Processing

These buttons control what should happen when the user inserts an <ENTER> into a text field. You can choose whether you follow a Windows convention of Carriage Return/Line Feed or just add the Line Feed character.

20.4.6. Default Codepage

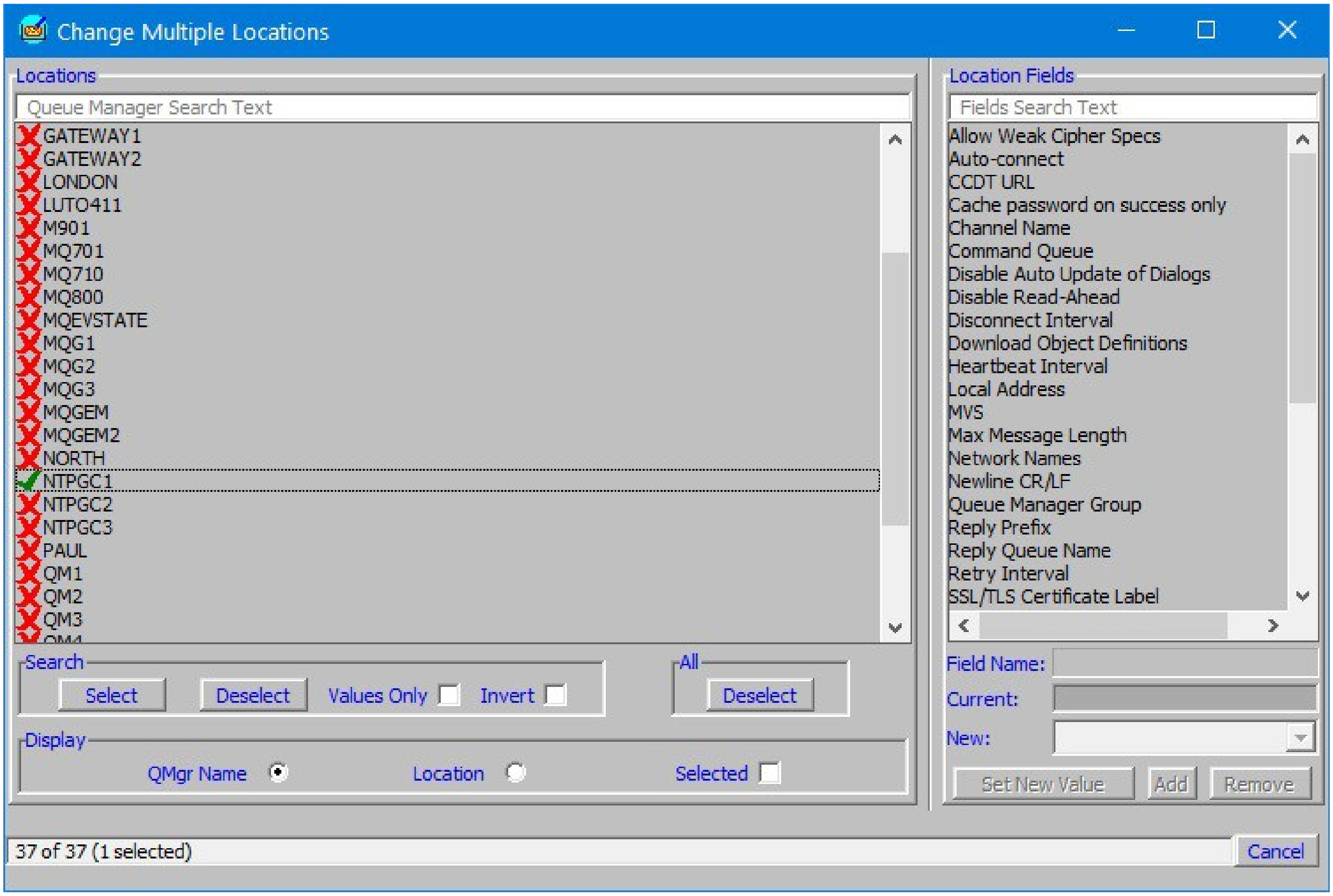
These fields show what the default codepage settings are for this location. The 'Default' codepage will be the codepage that is used when you press 'New' to create a new Message Descriptor. By default MQEdit will use the Queue Manager codepage which is will inquire when it connects. However, if required, you can put a value to use in the 'Preferred' setting.

20.4.7. Default Encoding

These fields show what the default encoding settings are for this location. This defines the encoding that is used when you press 'New' to create a new Message Descriptor. The 'Assumed Queue Manager' encoding is the value MQEdit assumes from the message responses. However, it could well not be correct since message responses from the Queue Manager have been converted. Either way, you can set the encoding you wish to use for new messages.

Chapter 21. Multiple Location Change Dialog

There can be times, particularly if you have a large number of location definitions, where you want to make the same change to many different locations. For example, you want them all to use the same Local Address or perhaps you want to add a group of locations to a particular Network Name. To do this the easiest way is to bring up the “Change Multiple Locations” dialog. This is available as a menu option under the *File* menu. By pressing that menu you will get a dialog such as the following:



The dialog is essentially split into two halves. On the left you have the locations to be changed and on the right you can select the field you would like to change. The dialog is fairly intuitive but we shall describe each field you have:

- **Queue Manager Search Text**
  You can type in a text string here and locations containing that string will be shown. The fields which are searched are:
    - Queue Manager Name
    - Location Name
    - Queue Manager Group
    - Network Names
    - Security Group Name

  If just a string is entered for example **MQ** then MQEdit will do a wildcard search as though ***MQ*** was entered. However, if you want to do a complete word search then you can enter the text in quotes, eg “**MQ**”

  If any locations are selected but they **do not** match the search string then they will still be shown but below a black bar in the list. This ensures that all the selected locations are always displayed and it is clear what locations will be affected.

- **Queue Manager Locations List**
  Here are listed the locations that can be acted upon. Essentially any field change will only be applied to the Queue Manager locations which are selected. The selection state can be changed just by clicking on the location and is displayed by either a tick or cross.

The search string above can be used to limit the locations shown and make finding locations easier. There are also various fields below which can make finding the locations you want easier.

- **Search**
  - **Select**
    This button will select all locations which match the search criteria
  - **Deselect**
    This button will deselect all locations which match the search criteria
  - **Values Only**
    Selecting this option will show only those locations which have a value for the selected field.
  - **Invert**
    Selecting this option will invert the current search criteria
- **All**
  - **Deselect**
    This button will deselect all locations
- **Display**
  - **Qmgr Name**
    This option will show locations by their Queue Manager Name
  - **Location**
    This option will show locations by their location name.
  - **Selected**
    Selecting this option will only show currently selected locations
- **Field Search Text**
  This field allows you to reduce the number of displayed fields by entering a simple string. Only fields either containing the string or which are related in some way will be shown. To that end there are certain keywords that can be entered which will match the fields. The list of keywords are:

connection	Fields related to making a connection to the server
channel or **chl**	Fields which are part of the local channel definition
ccdt	Fields related to the CCDT file
security	Fields related to security
command	Fields related to send commands to the Queue Manager
export	Fields related to exporting definitions

- **Field Selection**
  Here is a list of all the fields which can be changed 'in bulk'. Not every field is available since many of them don't make sense. For example, you can't change the 'Connection Name' since the expectation is that not many locations will exist at the same IP address and port combination.

  Selecting a field will have a number of effects:
  - The field name will be put in the 'Field Name' field
  - The Dialog title will contain the selected 'Field Name'
  - The current value field will show the current value of this field for the selected locations unless those definitions are different in which case it will display ***<Multiple Values>***
  - The current value for the field for each location will be shown in the left-hand list.
  - The new value drop-down is populated with the values of this field across all the Queue Manager.
- **New**
  Here you can enter the required new value for the field. The drop-down will contain all 'known' values.

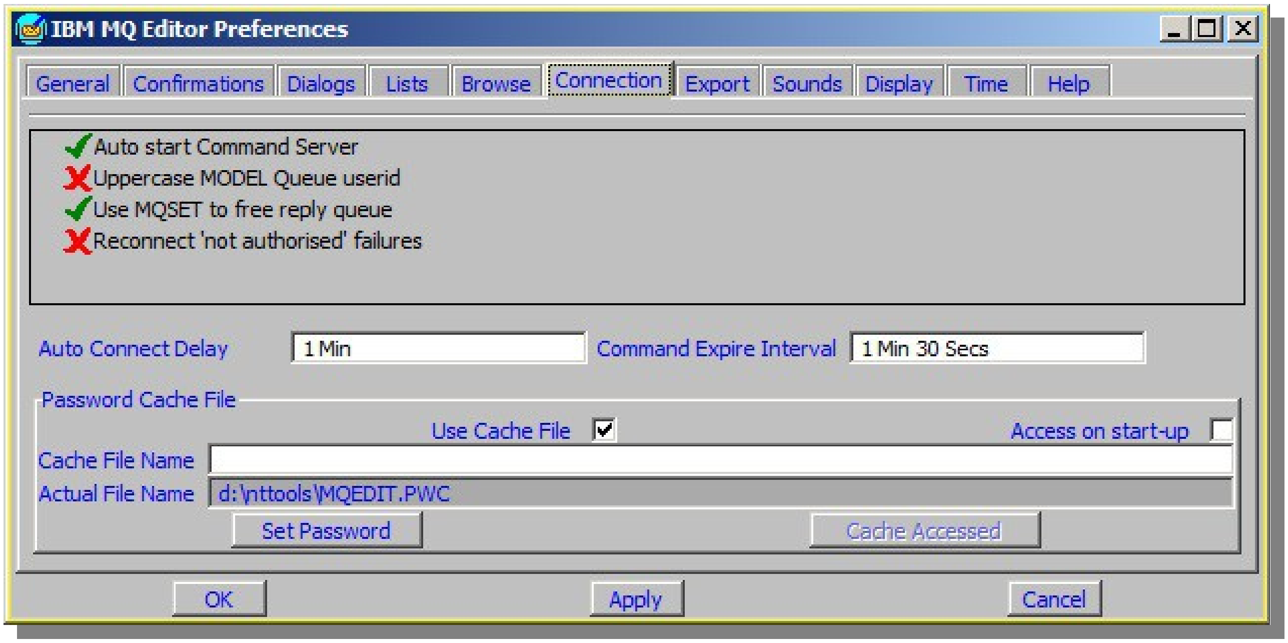
- **Set New Value**
  Pressing this button will change all selected locations to have the new value. This change will be reflected in the left-hand list and also any open location or client channel dialogs.
- **Add**
  Some values are comma separated lists. For example Network Names. For these fields the 'Add' and 'Remove' buttons are available and instead of replacing the entire definition pressing the add button will add the New value to the comma separated list if it is not already present. The New value is always added to the end of the list.
- **Remove**
  This button will remove the New value from any comma separated lists if present.

Chapter 22. Password Cache File

Since Version 6.0 MQ has had the ability to pass a userid/password combination across the connection. However, in Version 8.0 MQ will actually validate the userid/password combination against either the OS credentials or an LDAP repository. This means that providing a userid/password combination at connect time is now an easy and viable way of validating the user to MQ. However, it also means that, potentially, the user has to supply a set of passwords each time they run any applications, including MQEdit. To mitigate this problem MQEdit provides a password cache file. This is a file which contains the userid/password combinations for all the locations. These passwords are stored in encrypted form. Access to the password cache file itself requires a password. With a password cache file configured the user need only 'logon' to the cache file and all the credentials for the locations can then be provided automatically from the cache file.

22.1. Configuration

In order to use a password cache file you must tell MQEdit to use the file. You can then set, on a per location basis, which of the locations should store their passwords in the file. The setting of the cache file name is done in the Connection tab of the Preferences Dialog. On this tab we see a number of fields related to the cache file in a box labelled 'Password Cache File'.



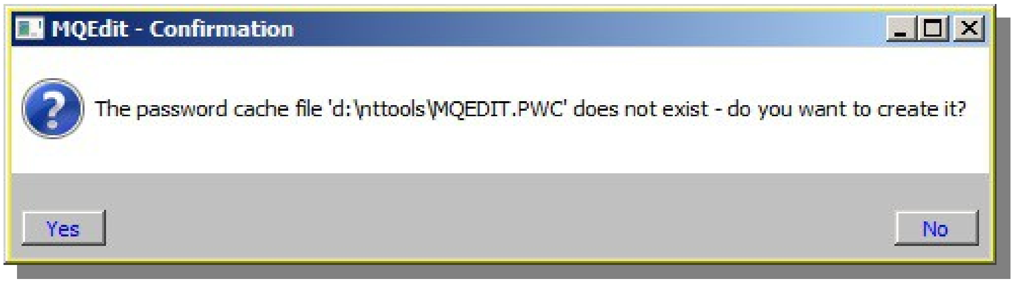
✓ Auto start Command Server
✗ Uppercase MODEL Queue userid
✓ Use MQSET to free reply queue
✗ Reconnect 'not authorised' failures

Auto Connect Delay: 1 Min
Command Expire Interval: 1 Min 30 Secs

Password Cache File
Use Cache File ☑
Access on start-up ☐
Cache File Name:
Actual File Name: d:\nttools\MQEDIT.PWC

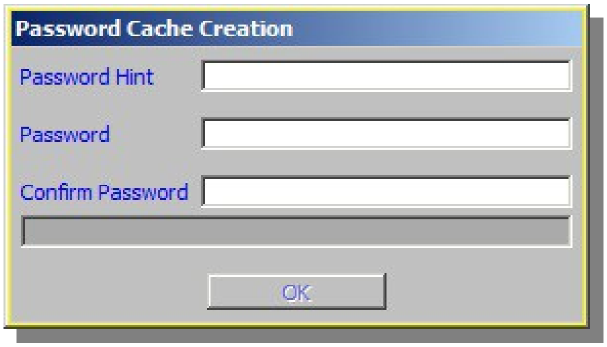
The two key pieces of information are a check-box indicating whether MQEdit should use the cache file, and the name of the cache file itself. The cache file name will default to a file called 'MQEDIT.PWC' in the same directory as the MQEdit configuration file, however, if you wish you can specify an alternate file name to use in the entry field. You could just simply give the full file name you wish MQEdit to use, or see Specifying the Cache File Name below for other options.

By default MQEdit will access the cache file only when it needs to. For example, when connecting to a location which has a password stored in the cache. However, by checking 'Access on start-up' MQEdit will prompt the user for the cache password as soon as you start MQEdit. This has the advantage that it “gets it out of the way”. If you specify a cache file that doesn't currently exist MQEdit will try to create it for you. Note that MQEdit will create a file but it will not create the directory structure. If you wish to use a file in different directory it will be necessary for you to create the directory if it doesn't already exist. This is done so that you create the directory with the right security access. Bear in mind that the password cache file contains your Queue Manager password so you want to store the file in a secure location that, ideally, only you have access to. So let's assume this is the first time that we are using a cache file. MQEdit will present the message shown below:



The password cache file 'd:\nttools\MQEDIT.PWC' does not exist - do you want to create it?

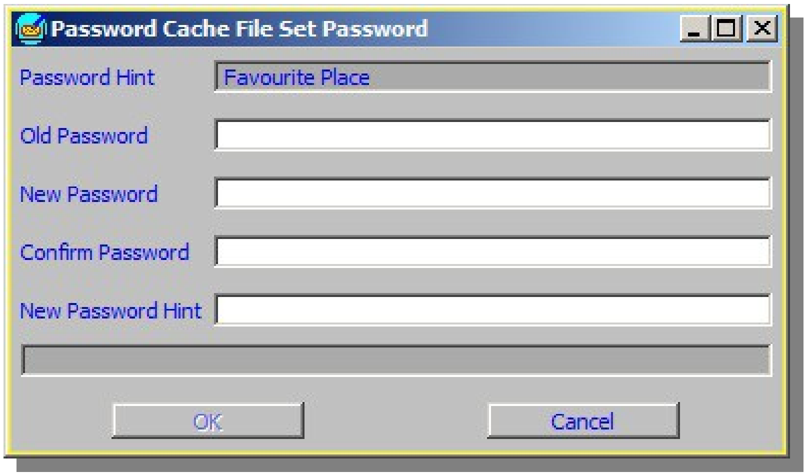
This alerts the user that you really are creating a new file. If you select 'Yes' then MQEdit will present the dialog below:



Since this cache file does not currently exist you have the opportunity to set the password for the file. MQEdit also allows you to set a password hint if you wish. This can be useful if you have lots of password you need to keep track of. You can provide a clue that should only mean something to you but gives you a reminder as to what your password for this file is. It goes without saying that you should not make the hint too obvious or guessable.

Anyway, fill in the fields you wish and press OK. If you did it right the dialog disappears and the Preferences dialog now shows the button as 'Cache Accessed'. This indicates that MQEdit currently has access to the cache. The cache file itself has now been created and will exist in the file system. By all means go and have a look at it. If you are interested in the file format then take a look at Cache File Format on page 125 which describes a little of how the file is constructed.

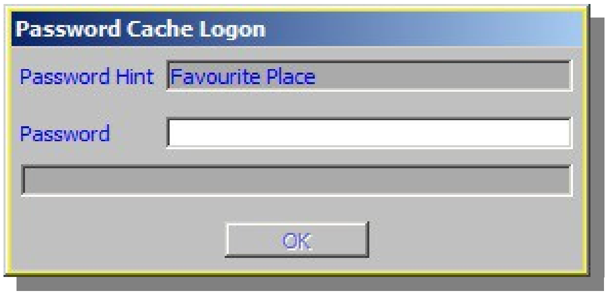
If at any time you wish to change the password of the cache file then you can press the 'Set Password' button. This will show a dialog like the following:



Password Hint: Favourite Place

This dialog requires that you provide the cache password and gives you the opportunity to provide a new password and a new hint for the file. As with most password mechanisms it is advisable to change the password on your cache file on a regular basis. MQEdit will not force you to change the password at any interval.

Once the cache file has been created, then when MQEdit has need to access the file it must be given the same password. You can either pass the password in as a parameter to the program (see parameter **-P** in MQEdit Parameters) or you will be shown the following dialog:



Password Hint: Favourite Place

MQEdit will give you four attempts to get the password right. If after four attempts you still have not provided the correct password then the cache file functionality will be suspended and you will have to enter all location passwords manually. The only way to re-instate access to the password cache is to end and restart the MQEdit application.

22.2. Specifying the Cache File Name

By default MQEdit will use a file called MQEDIT.PWC in the same directory as the MQEdit configuration file. However, there are times where you might wish to change this location. For example, you might wish to have a specific directory with very stringent access rights. Another reason may be that you share the same configuration file for multiple users but wish each user to have a different file and/or directory. You can achieve these by entering your own file name in the given entry field. However, as you may expect it may not be quite as simple as just typing the file name you wish to use. The rules MQEdit follows is:

- If the path is not specified then the MQEdit configuration path is used
- If the file name is not provided then MQEDIT.PWC is used
- Any sequence of characters %XXX% will be replaced by the value of environment variable XXX

As you type a value into the entry field you will see that the value of the 'Actual File Name' change. So, at any time you can see the exact file name that MQEdit will use. For example to give each user their own cache file in their own directory it may be as simple as specifying a value of %USERPROFILE% depending on what environment variables you have defined.

22.3. Cache File Format

The cache file is just a normal OS file. You can look at it in 'notepad' or load it up into your favourite editor. However, perhaps not surprisingly you will not see the passwords of either the cache file itself or the Queue Manager locations in the file. Instead these values are encrypted. MQEdit uses two mechanisms. The cache file password uses a one-way hash algorithm to construct an authorisation code. The authorisation code of the original password and any given password at access time must match.

The passwords for the Queue Managers themselves use a different encryption algorithm that needs to be reversible so that the password can be passed to MQ. The key for this encryption is based on the cache file authorisation code. So, the same Queue Manager password will be encrypted differently depending on the cache file password itself. The fact that the password encryption is reversible is is one of the reasons why it is recommended that you change the password on your cache file from time to time. Note that the encryption of the passwords uses a non-trivial encryption routine but, like any encryption mechanism, it could be decrypted using a brute force attack and sufficient time. It is therefore recommended that you take the additional precaution of ensuring that your password cache file is protected from prying eyes as much as possible. Put the file in a protected directory. When MQEdit creates the file it will make the file non-shareable by default which should mean that only the current user and machine administrators have access to the file.

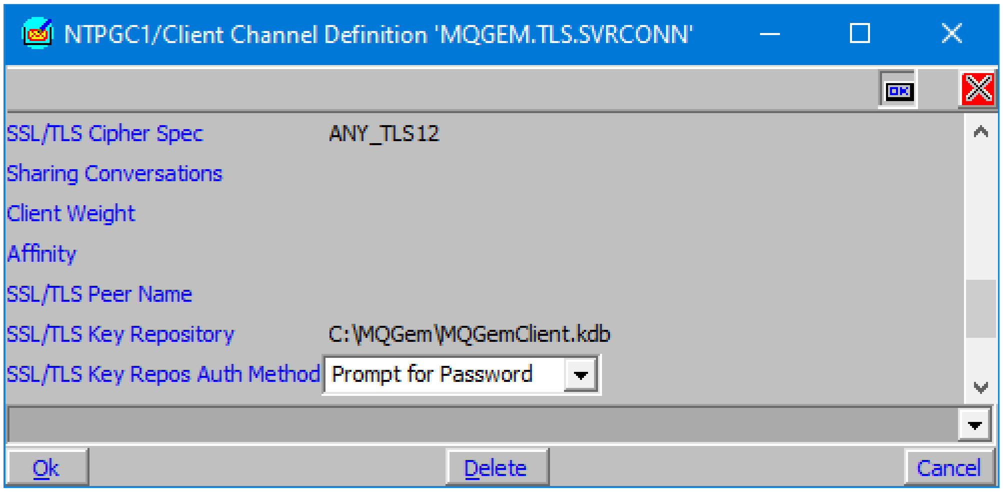
The file format itself is fairly straightforward. You can if you wish edit the file manually as long as you follow a few simple rules.

- Comment lines may be added a anywhere in the by starting the line with an asterisk (*) character.
- Blank lines may be added anywhere
- The first two 'active' lines in the file must be the AuthCode and Hint lines
- Queue Manager lines must be on a single line with fields Qm, Q, User, Pwd in that order.

So, if you wish to add comments to the password file then by all means do so. Another thing that may be convenient is to delete passwords for certain locations. MQEdit does provide a 'Delete Password' button in a location dialog but if you wished to delete the passwords for, say, twenty locations it may be quicker to just edit the file and remove the lines you want. Bear in mind that if you do choose to edit the file then you should try and ensure that MQEdit is not trying to access the file at the same time. Of course at any time you can always just delete the file and re-create it. Clearly the user would then be prompted to re-enter any Queue Manager passwords as required. These would then be stored in the a newly created password cache file.

22.4. Key Database File Password

IBM MQ V9.3.0 removed the reliance on a stash file in order for the MQ Client to know the password to the Key Database (KDB) file and allowed the password to be provided on the connect call. To use this in MQEdit, configure your client connection from the location dialog and ensure your KDB file name is provided in the "SSL/TLS Key Repository" field; then choose "Prompt for Password" in in the "SSL/TLS Key Repos Auth Method" field. The alternative choice in this field "IBM MQ Configured" assumes that your KDB password will be found by the MQ Client in either the MQKEYRPWD environment variable; the mqclient.ini file field SSLKeyRepositoryPassword; or a Stash File.

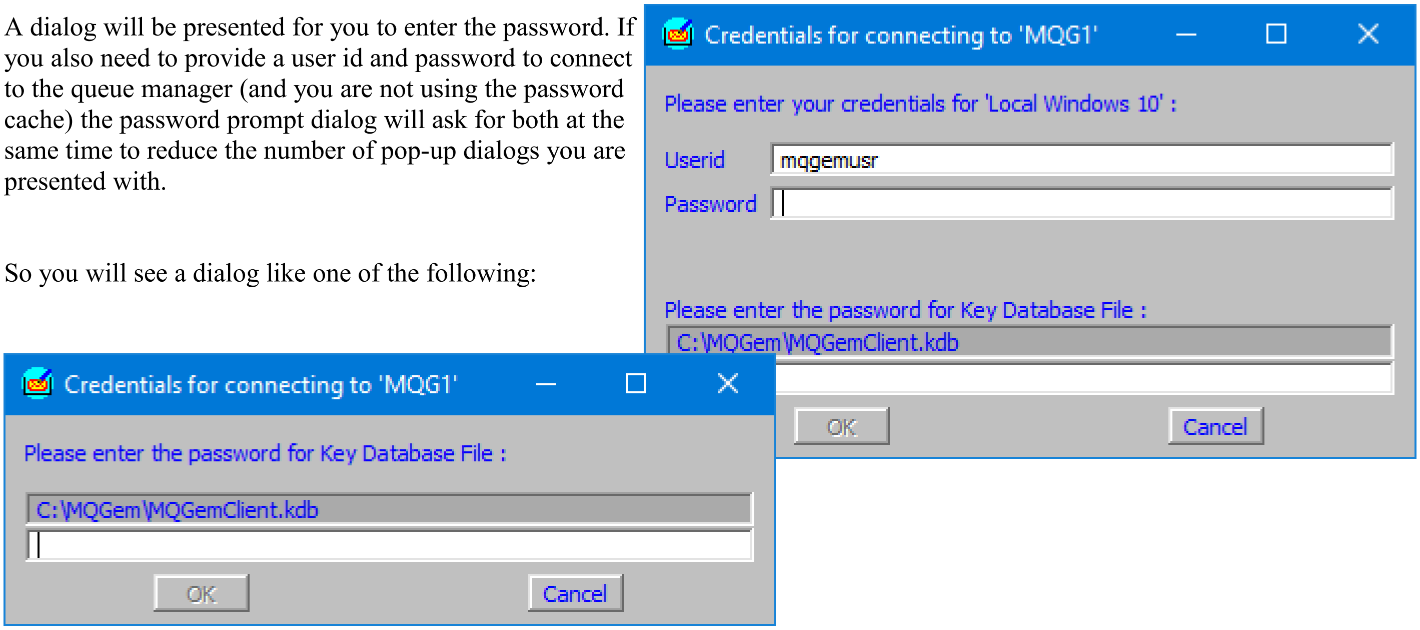

NTPGC1/Client Channel Definition 'MQGEM.TLS.SVRCONN'

SSL/TLS Cipher Spec: ANY_TLS12
Sharing Conversations
Client Weight
Affinity
SSL/TLS Peer Name
SSL/TLS Key Repository: C:\MQGem\MQGemClient.kdb
SSL/TLS Key Repos Auth Method: Prompt for Password

Using the "Prompt for Password" method will cause MQEdit to prompt you for the password when it is needed. The password will be cached in memory for the lifetime of the program, and all queue manager connections specifying the same KDB file name will use the same remembered password.

A dialog will be presented for you to enter the password. If you also need to provide a user id and password to connect to the queue manager (and you are not using the password cache) the password prompt dialog will ask for both at the same time to reduce the number of pop-up dialogs you are presented with.

So you will see a dialog like one of the following:


Credentials for connecting to 'MQG1'

Please enter your credentials for 'Local Windows 10' :

Userid: mqgemusr
Password:

Please enter the password for Key Database File :
C:\MQGem\MQGemClient.kdb

Credentials for connecting to 'MQG1'

Please enter the password for Key Database File :
C:\MQGem\MQGemClient.kdb

If you change the password of the KDB file and don't want to cycle MQEdit, you can use the "Forget KDB Password" button found on the Security tab of the Location dialog (see 20.3.10. Forget KDB Password on page 118) to clear what MQEdit is remembering and force the password prompt dialog to be used for the next connection that needs it.

Chapter 23. Queue Manager Navigation Pane Usage

The Queue Manager navigation pane shows all of the configured Queue Managers and downloaded queues in simple list display. Elements of the control can be hidden or shown by clicking on the arrows. You can also choose which Queue Managers are displayed using the “Search Locations“ field which is displayed when you select the main window “Search Locations“ menu described on page 102.

The display can be driven either by mouse or keyboard. For keyboard control the item in focus is shown either with a slightly different background colour or with a focus rectangle around the text.

23.1.1. Keyboard Control

The following keys can be used to navigate the container:

Key sequence	Action
Arrow Keys	To move the focus from one item to another.
Space	To activate a button, selectable text or a sub tree arrow
Home	Move to top of container data
End	Move to end of container data
PageDown	Move one page down through container data
PageUp	Move one page up through container data
Shift+Home	Move to the far left of container data
Shift+End	Move to the far right of container data
Shift+PageDown	Move one page to the right in the container data
Shift+PageUp	Move one page to the left in the container data
F9	Toggle Expand/Contract state of current line
F9+Shift	Expand current line and all sub trees of current line
F9+Ctrl	Contract current line and all sub trees of current line
F9+Alt[24]+Shift	Expand all lines in container
F9+Alt+Ctrl	Contract all lines in the container

---

[24] Depending on keyboard mapping it may be necessary to use the ‘Alt Gr’ button rather than just the ‘Alt’ button.

Chapter 24. Reply Queue

24.1. Reply Queue Details

The editor uses a reply queue to receive all of its reply messages from a Queue Manager it connects to. By default the name of this queue is SYSTEM.DEFAULT.MODEL.QUEUE but can be overridden by changing the Reply Queue name in the location definition. The local or model queue of whatever name you choose must already exist on the respective Queue Manager.

If more than one instance of the editor is run against the same Queue Manager then each instance must use a different local reply queue or a model queue.

24.2. Model Reply Queue

Using a model reply queue is useful if many instances of MQEdit will be run against the same configuration file at the same time. The disadvantage of using a model queue as the reply is that an instance of MQEdit has no 'fixed address'. In other words the reply queue name is not predetermined.

When a model queue is used as a reply queue the actual queue name used is governed by the DynamicQName specified in the object descriptor. The Reply Prefix field of the location dialog allows the user to specify the value for DynamicQName field.

The field can be any sequence of characters providing they are valid Queue name characters. Two special characters have special significance:-

* An asterisk at the end of the string indicates that MQ should add a sequence of hex characters to ensure that the Queue name is unique.

%u This sequence indicates that the current userid should be inserts at this part of the name.

If the reply prefix is empty the value of ***MQEdit.%u.**** will be used. On my own machine using this default setting, with a userid of CLARKEP, the queue names I get are of the form **MQEdit.CLARKEP.412EE8F603370020**

Chapter 25. Security

At it's heart MQEdit is an MQI application, and it purely uses the MQI to communicate with the Queue Manager. Consequently it is secured using the normal MQ security mechanisms. Giving a user access to MQEdit does not add to their access rights; if they can change the contents of queues with MQEdit then they could add and remove messages from queues using other programs. It is worth just saying a few words about each aspect of security.

25.1. Connection

IBM MQ offers many different ways to secure your connection and, over the years, the choices have increased significantly. With MQEdit it is possible to secure your connection in whatever way suits your installation.

25.1.1. Authentication

There are a number of different ways of authenticating your connection such as channel exits, SSL/TLS and userid/password. Since it's introduction in MQ V8 userid/password has become very popular, and you can supply a user id and password for the connection made by MQEdit. MQEdit makes this even easier with the ability to store your passwords in a password cache file. For more information see “Password Cache File“ on page 123.

25.1.2. Wire Security

You should also consider how secure your communications are. If you have strong physical security then it may be acceptable for messages to flow 'in the clear' however if you wish to hide the contents from prying eyes then you should use an SSL/TLS CipherSpec to ensure that the messages are not readable on the wire.

25.2. Queue List

When MQEdit queries the queue list the Queue Manager will only send the list of queues that the user is authorised to see. So, it is entirely possible that the list of queues shown in the MQEdit window is not the complete list. This is a useful aspect of MQ and means that you can partition the queues according to your users authorities.

25.2.1. Browse Only

IBM MQ allows you state, on a per queue basis, whether a particular user is allowed to put messages to a queue. For example, it is common that a user would not be able to put to queues like SYSTEM.CLUSTER.TRANSMIT.QUEUE. However, just because messages can not be changed does not mean that MQEdit is not still useful. Being able to browse the messages on various queues in an IBM MQ installation can still be very handy and MQEdit has an excellent message formatter which allows the user to interpret those messages easily. It also has features such as unload, find, and copy&paste which may allow an Administrator to diagnose a problem easier. Consider for each queue whether update, browse or no-access is most appropriate for each user or user group.

25.3. Message Security

25.3.1. Message Context

In addition to the message content there are various context fields in the Message Descriptor. These fields contain such things as the user and application that put the message. The lowest level of MQ authority does not allow the user to provide these fields and they will be set by the queue manager to the values in effect when the MQPUT is made. If you want to be able to specify these fields then clearly you need sufficient authority to “set context” as it is known.

25.3.2. Advanced Message Security

If you are using Advanced Message Security (AMS) then messages will be encrypted between source and target. It follows, therefore, that these messages will not be formatted unless the user running MQEdit is identified as a valid recipient.

Chapter 26. General Actions

26.1. Dates and Times

Dates and Times in MQ, such as the time a message was put, have a number of problems.

- The time is displayed in two separate fields.
  This makes sorting in lists very awkward because sorting by either field doesn't yield a time sorted list.
- The format is fixed and not always easy to read.
  For messages it is something like: Date: 20190528 Time: 06431570

Of course everyone has their own preferred way of dates and times being displayed. One of classic confusions is whether dates are YYYYMMDD or YYYDDMM however it goes far beyond that. MO71 allows you to have pretty much whatever format you prefer.

MQEdit removes these problems by allows you to refer to a combined date/time field rather than the individual date and time fields. These original date and time fields can still displayed if you wish but you now have the added field which will display the date and time in any format you wish.

To configure this, in the 'Time' tab of the Preferences Dialog you specify the format of time you wish to see.

The other choice you can make is whether these combined times should be shown to you in your local time, or as the Queue Manager sent them to MQEdit. This is done by checking the 'Show local times' checkbox in the the 'Time' tab of the Preferences Dialog. In the message list, the combined Date&Time fields have their own type of column filter. This filter allows you to filter the list in various ways such as 'until a particular time' or 'from a time for 2 hours' etc. For an explanation of the options available please refer to 'Date&Time Column Filters' on page 30.

26.1.1. Recent Dates

MO71 also allows you to highlight dates which occurred recently. Very often, if you are tracking down a problem, it can be a recent message which caused an issue. It can be useful therefore to highlight dates which happened recently. MQEdit allows you specify a different time format for times which happened either in

- the last 24 or 48 hours
- or Today or Yesterday

In the first case you are only interested in the elapsed period of time. In the latter case you are interested in the demarcation of days. You might for example have a format which says “Today %TT” which, rather than having the date, makes it very clear that this particular message was put today.

26.2. Splash Screens

Splash screens are windows displayed to the user, triggered by a program event, to notify the user of information. What is displayed to the user is entirely up to you. You can display text, pictures, even a web page! Splash screen settings are configure in the 'General' tab of the 'Preferences Dialog'. There are currently two events that can cause a Splash Screen; when MQEdit initially starts and when MQEdit connects to a Queue Manager location. Splash screens can be used for a variety of reasons.

- **To give users general information about the system and ownership**
- **To notify users of the type of Queue Manager**
  For example, you might have a splash screen on all your production Queue Managers to alert the user that this is a production system and that any changes should be treated with care.
- **To notify users of outages and maintenance schedule**
  Suppose you need to bring a queue manager down for maintenance. You could have a splash screen which would tell any user why the system is down and when it is likely to come back. This could even prevent phone calls in to your department of people complaining that they can no long connect to their favourite Queue Managers!

Because of these different scenarios, the configuration of the Queue Manager Splash screen can be given in two parts. We have the notion of a 'Normal' Splash screen and a 'Specific' Splash screen. The idea is that the 'Normal' splash is the screen that is normally shown to the user – this would just give general information about the Queue Manager, for example its type (Production/Test/Development), its maintenance schedule and its owner. Then we have the 'Specific' Splash screen which is shown to tell the user of specific, more unusual events. For example that the Queue Manager is currently down and the expected time it will be restarted.

Whenever MQEdit is about to display a Queue Manager Splash screen it will look for a specific Splash screen first and only if it doesn't find it will it display the Normal Splash screen. Of course you may decide that you don't wish to use one or other type of Splash screen and that is perfectly fine. If there is no Splash screen of that type then MQEdit will ignore it and use the other one.

Splash screens are shown to the user in response to a user action. They will not be shown just because a Queue is auto connecting but as the result of the user doing something. Once a Splash screen has been shown for a Queue Manager it will not be shown again until connection is ended.

Configuring a splash screen is very simple. All you need to do is put the location of the Splash file in the appropriate entry field. The splash file can be a number of different formats.

Splash Type	Example	Why that type?	Notes
Text file	c:\MQEdit\splash\startup.txt	Ends **.txt**	
RTF file	c:\MQEdit\splash\startup.rtf	Ends **.rtf**	Pictures are currently not supported.
HTML file	c:\MQEdit\splash\startup.html	Ends **.html** or **.htm**	JavaScript is not currently supported
Remote Web Page	http://www.mqgem.com/MQEdit/startup.html	Starts **http://**	JavaScript is not currently supported

In each case MQEdit will attempt to show the contents of the file to the user in s window. Bear in mind that the intention is that these windows of information should be kept small and concise. If necessary you can have links to other information which can be displayed in a browser.

If a file does not exists (or is not accessible) then no error message is displayed, it is just that the user does not see a splash screen. In this way the administrator can control, centrally, whether a splash screen is shown or not. For example, suppose the department has a central file server. You could configure all of your MQEdit users to use a splash startup file of 'h:\central\MQEDit\splash\startup.rtf' and, let's say, normally this file doesn't exist. However, periodically you could add the file, perhaps to tell users about a maintenance window the following weekend, and so the next time the user starts MQEdit they will be told about the scheduled maintenance.

26.2.1. Configuration

Configuration of the Splash screens is done in the General tab of the Preferences Dialog described on page 105. On this dialog there are three entry fields which allow you to enter a Splash screen of each type. At this point you may be wondering how you can enter a single value in the Preferences Dialog which produces a different result for different Queue Managers. The answer is that you can use inserts in the file name.

Two inserts are available:

Insert	Example	Effect
%m	c:\MQEdit\splash\%m.rtf	The %m insert will be replaced by the location Queue Manager name
%g	c:\MQEdit\splash\%g.html	The %g insert will be replaced by the locations Queue Manager group name This makes it easy to have a different Splash screen for each logical group of Queue Managers

There is a test button to the right of each setting. This allows you to quickly verify that the Splash screen file is specified correctly and the contents is shown. For the Queue Manager splash screen dummy values of 'mqgemqm' and 'mqgemgroup' are used for the Queue Manager name and group name inserts. If you wish to see what the Splash screen looks like for a particular location then you can press the 'Show Splash Window' button in the Options Tab tab of the relevant Location Dialog described on page 118.

26.2.2. Location and Size

By default Splash Screens are shown in the centre of the monitor and are half the size of the screen. The user can re-position and resize the window but these new values are not remembered by MQEdit since the next time the window is displayed it could contain completely different content. However, if required you can set the preferred size in the configuration. The way you do this is to simply add the notation **[*xsize*, *ysize*]** to the end of the file name.

For example **c:\MQEdit\splash\%g.html[1000,600]** requests that the window is initially sized at 1000 pixels by 600. If a size is set to zero then the default size for that dimension will be used. If a size is given that is larger than the size of the screen area then the screen are size is used. So, for example, **c:\MQEdit\splash\%g.html[99999,99999]** will display the window and completely fill the monitor.

26.3. Master Configuration File

Some MQEdit installations consist of a central department of core MQ experts and a number of satellite departments who are aware of MQ but have wider responsibilities. In these installations it can be handy to have the basic MQEdit configuration done centrally in terms of add location definitions and ensuring that their connection parameters, such as the channel definition, are correct. However, you still want the users of MQEdit to be able to have their own settings, such as filter, dialog positions etc, remembered across restarts so you don't want to be too prescriptive with the configuration file you give them.

The solution is to use a master (or joint) configuration file. This essentially allows you to import central location definitions from a master MQEdit configuration file each time MQEdit is started. For example:

```
mqedit -j h:\MQ\central\MQEdit.cfg
```

This command tells MQEdit to merge any location definitions it finds in this central file with its own definitions. Any new locations are added and any existing locations are modified to match the central file's connection parameters. Note that not every parameter associated with the location is migrated, they are mainly those related to the actual connect. The list of fields migrated are:

- Location
- Group
- Network Names
- Via Qm
- Reply Queue
- Reply Prefix
- Command Queue
- Server Queue
- CCDT URL
- Client and any Client Channel Definition
- MVS
- Auto Connect
- 'Userid and Password' connect option
- 'Security Exit Only' connect option

26.4. Licence Command

In most MQEdit installations the administration of the software licences is done centrally and the licence is copied out to the relevant machines where MQEdit is actually run. To make this easier and more automatic MQEdit allows the user to specify a command which can run on the machine that will do the copy for you, this is known as a licence command. MQEdit will run the command if it detects that there is no valid licence, or if the current licence has less that 30 days left to run.

The command itself is configured in the General tab of the Preferences Dialog. For example we could specify something like:

d:\nttools\getMQEditlicence.cmd %d

The **%d** is replaced by MQEdit with the current configuration directory.

The contents of the command file itself can be whatever you choose but in general it would be something pretty simple such as a copy command. For example:

```
copy h:\licences\MQEdit\mqgem.lic %1
```

So, with this combination whenever MQEdit invokes this file a copy of the latest licence from **h:\licences\MQEdit** will be copied to the MQEdit program directory. In other words, provided there is valid licence in this directory then MQEdit will pick it up and start using it. Of course the command file could be more complicated. You could, for example, make an FTP call to retrieve the file from a central server. Or you could send an email to alert a user of the situation.

26.4.1. Command Inserts

There are two command inserts which can be used

Insert	Replaced with

%d	The configuration file path (the default location for the licence file)
%r	The number of days remaining on the current licence

26.4.2. Command File Exit Code

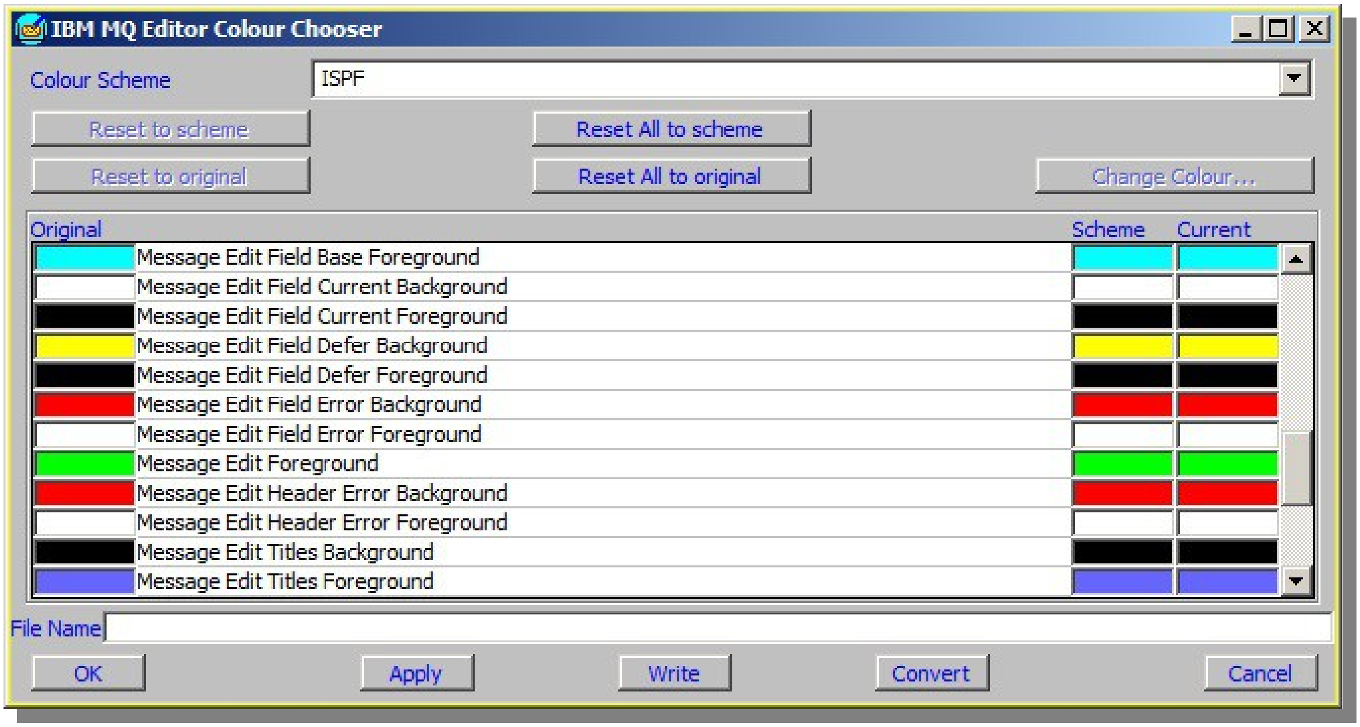
If the invocation of the command fails or it returned a non-zero exit code then MQEdit will display a message box to explain the situation. To deliberately return a bad exit code use the **exit** statement. For example:

```
exit 12
```

Because of the way Windows processes program exit codes command files should not return an exit code of 259.

Chapter 27. Colour Dialog

The colours of most of the elements of the application can be configured in the colour dialog available under the ‘View’ menu in the main window.



Colour Scheme: ISPF

Original		Scheme	Current
🟦	Message Edit Field Base Foreground	🟦	🟦
⬜	Message Edit Field Current Background	⬜	⬜
⬛	Message Edit Field Current Foreground	⬛	⬛
🟨	Message Edit Field Defer Background	🟨	🟨
⬛	Message Edit Field Defer Foreground	⬛	⬛
🟥	Message Edit Field Error Background	🟥	🟥
⬜	Message Edit Field Error Foreground	⬜	⬜
🟩	Message Edit Foreground	🟩	🟩
🟥	Message Edit Header Error Background	🟥	🟥
⬜	Message Edit Header Error Foreground	⬜	⬜
⬛	Message Edit Titles Background	⬛	⬛
🟦	Message Edit Titles Foreground	🟦	🟦

The colour dialog displays all the dialog elements in a list and a number of action buttons. The list of elements shows the colour of element when the dialog was started, a text description of the element, the colour of the selected scheme for this element and the current colour of the element that would be applied if the ‘OK’ or ‘Apply’ button was pressed.

The element list allows multiple selection using the standard extended selection scheme. The action buttons will either apply to those items selected or to all the elements in the list.

For example, to change to use the Grey colour scheme use the following sequence.

1. Select the Grey scheme in the colour scheme drop down list
2. Press ‘Reset All to scheme’ button
3. Press ‘Apply’ button

27.1. Colour Schemes

Colour schemes are a fast and simple way to set all the colours in the application to a predefined set of values. When the dialog is displayed the colour scheme which most closely matches the colours you have selected will be preselected. You can use the colour scheme as it is provided or use it as a base and make a few selections to certain fields.

If you create a colour scheme which is significantly different than the provided set of schemes then by all means send me your scheme so it can be included in the next release of the program. To do this run the program with the -d flag, for example **'mqedit -d'**. In the colour dialog you will now see a **'write'** button. By pressing this the program will write a file called **c:\temp\schemes.txt.** Please send me this file, together with the name you'd like to use for the scheme. Here is an opportunity for you to get directly involved in the development of MQEdit!

27.2. Extended Selection

By holding down the **‘Alt’** button at the same time a selection is made from the colour list the application will select (or deselect) not only the particular item in question but all the items in the colour list which have the same current colour. This allows the user to easily change all the items that use the same colour to now use a different colour. On some systems the two **‘Alt’** buttons behave slightly differently. One is additive to the current selections and the other will replace any previous selection.

Chapter 28. Text Inserts

In various places you can specify text strings which contain inserts. This means that a fixed string can show more dynamic content, for example, such as the current date. The general mechanism is that inserts are identified by specifying a '%' character followed by one or more characters denoting the type of insert. However, note that a new line is signified using a backslash (\) character.

The available inserts are as follows:

Insert	Meaning	Conditional	Notes
%c	Current Command Name	Yes	
%m	Queue Manager Name	Yes	
%l	Location Name	Yes	
%d	Day short form e.g. Sun, Mon, Tue		
%dd	Day e.g. Sunday, Monday, Tuesday		
%D	Day of the month e.g. 16		
%DD	Day of the month e.g. 16th		
%e	Elapsed time in H:MM format		Amount of time until now in H:MM format
%E	Day of the month always 2 digits		
%M	Month number e.g. 11		
%MM	Month short form e.g. Jan, Feb, Mar		
%MMM	Month e.g. January, February, March		
%Y	Year short form e.g. 03		
%YY	Year e.g. 2003		
%T	Time e.g. 22:18		
%TT	Time e.g. 22:18:35		
\n	New line		Export title only
\\	\		

28.1. Conditional Text Inserts

Not all of these values are available all the time. For example, depending on the context there may be no current Queue Manager and so it will have no current value.

Let's consider a print example. Suppose one wished to have a header that printed

```
Queue Manager: QMNAME
```

This would be fine for printouts that had a current Queue Manager but on the others you would just get :-

```
Queue Manager:
```

This is not very desirable. We need a way of switching off text if a variable is not present. The answer is to enclose the text in brackets. By specifying a header of '%m{Queue Manager :%m}' we are saying only print out the contents of the brackets if you have a value for %m.

Chapter 29. File Name Inserts

When exporting from a dialog you can specify the name of a file to write the data to. If you like you can specify inserts in the file name to make the file name more dynamic. For example, you can put the time of day or the date in the file name, The inserts are all specified following a % (percent character).

The inserts and their meanings are as follows:

Insert	Effect
q	Queue Manager name
l	Location name
i<*n*>	Cyclic index number For example **file%i3** will write to files file1, file2, file3, file1, file2…. etc
H	Two digit hour
M	Two digit minutes
S	Two digit seconds
d	Two digit day of month
m	Three character month name eg. Jan, Feb, Mar….
y	Four digit year
D	Three character day of week eg. Mon, Tue, Wed…
t	Simple time format eg. 18.14.03
%	Percent character

Chapter 30. Time Interval

A number of configuration options are some form of time interval. These parameters use a common input format. The format allows for a list of value and unit pairs. The units are milliseconds(ms), seconds, minute, hour, day, week. Generally speaking only the first letter of the unit need be used. The following inputs are examples of valid input.

- 1 day 3 hours 25 minutes
- 1d3h25m
- 35 ms
- 25

In cases where no unit is specified the default unit for the particular field is used. Note that not all units are valid for all field types. Generally speaking fields which have a default unit of milliseconds do not allow units larger than seconds. The user should always press the ‘Apply’ button and verify that the output field value matches the value which was required. Note that the output format will not match exactly the input but will be in the standard value. For example, entering **48h** will be output as **2 days**.

Chapter 31. MQEdit Parameters

The Editor program does not require any parameters but if you wish there are a number of parameters which can modify it's behaviours. Perhaps the two most useful capabilities are:

1. Passing in the location of the configuration file.
2. Passing in the password to the password cache file

31.1. General Parameter List

- **-E**<*Mini-dump File Directory*>
  This parameter instructs MQEdit to install an exception handler and to write any exceptions in the form of a min-dump file to the directory given. The **-F** parameter can be used to control the type of mini-dump generated.

- **-f <***Configuration Directory/FileName***>**
  This parameter can be either just a directory name or the full path to the configuration file to use. By default the program will store the configuration in a file called MQEDIT.CFG, in the directory where it finds the program. Using this parameter you can override this location and give a different directory and/or file name. Note that the directory specified will also affect the locations of the objects file (MQEDIT.DFN) and the authorities file (MQEDIT.AUT). A common usage of this parameter is to specify '**-f** .' which requests that all configuration is read and written to the current directory.

- -**F**<*Mini-dump file format*>
  This parameter controls what type of min-dump is generated whenever a dump is taken. The following parameters are supported:

Format	Meaning
normal	This format generates a normal mini-dump. A normal mini-dump contains only thread and stack information, no heap memory is included. A normal mini-dump is typically only about 50K but may not be sufficient to diagnose all problems. This is a default if no **-F** parameter is specified.
full	This format causes a full mini-dump to be generated. A full mini-dump file contains thread, stack **and** heap storage information. Consequently the file is much larger, often around 100MB. Full dump files should only be necessary in rare circumstances. Please do not send full dump files to service via email. Instead use an internet file share of some kind.

- **-j** <Master Configuration File>
  Allows you to specify the location of a master configuration file from which location definitions should be retrieved.

  Please see Chapter 26.3 Master Configuration File on page 132 for more information.
- **-m <***Initial Queue Manager*>
  The name of the Queue Manager that the editor should initially connect to.
- **-P<***Password Cache Password***>**
  The password to access the password cache file.
- **-q** <*Queue Name*>
  The name of the Queue that the editor should initially browse.
- **-r**
  Do not write to the CFG file, open for read-only.
- **-t** <*Message Token*>
  The message token, as a 32 character Hex string, that the editor should initially read for editing.

Chapter 32. Usage Tailoring

You can change the menu options available in MQEdit depending on the type of user which will be using MQEdit. This is known as Usage Tailoring. For example, suppose you want some users to be able to display the queue manager dialog but not others. You can also remove menu items, such as viewing the error logs. There are two main reasons why you might want to do this. Firstly because you don't want certain users to be able to use powerful features and 'potentially' cause problems. Secondly, you want to hide any superfluous complexity and present the user with a simpler interface.

Tailoring is achieved using a separate configuration file – MQEDIT.AUT (where .AUT stands for Application Usage Tailoring). This file must be created manually (using your favourite editor) and placed in the same directory as the MQEDIT.CFG file. If this file is not present then all options will be made available to all users. The MQEDIT.AUT file can be write protected since it is constructed and updated only by the administrator setting up the tailoring. Providing the application can read the file it can be protected as required. It should be noted that this mechanism is not intended to be fully secure; you are also recommended to make the corresponding authority configurations in your Queue Managers. For the majority of these kinds of situations, however, a simple way of preventing the full range of options being presented to the user is sufficient.

Object tailoring can either be specified additive, for example **location_create**, or subtractive, for example **!location_create**. The advantage of the subtractive it that it future proofs for future securities. For example, suppose you want a user to be able to do anything with a location except delete them then you could use the object tailoring keywords **location_all !location_delete**.

The best way of explaining the format of the file is to show an example:-

```
# Set Usage Tailoring for users of MQEdit
#
# Global Tailoring
qmgr_display queue_display location_display
nomenu_log_file

# Per Queue Manager Tailoring
QM: NTPGC1
     all
QM: NTPGC2
     location_change
QM: NTPGC3
     !qmgr_display
```

The format of the file is entirely free-format. White space can be added/removed wherever the user chooses. Comments can be added from the first ';' or '#' character to the end of the line.

The file is structured into global tailoring and tailoring on a per queue manager basis. The global tailoring appears before the first 'QM:' keyword. In addition to object tailoring you can specify explicit keywords which will remove items from the various menus in MQEdit. This can make MQEdit much simpler to use, although it naturally reduces the available functionality.

The 'QM:' must be followed by the name of the Queue Manager. This Queue Manager name may contain wildcards characters '*' and '?'. If wildcards are used then the following values will apply to all Queue Managers which match the value. The Queue Manager name itself is followed by zero or more tailoring keywords. Any tailoring keywords specified in the Queue Manager section are additive to the global authorisations. They do not replace the tailoring. So a user running with this file can:-

- Do all operations against NTPGC1
- Display queues and display and change the location for NTPGC2
- Display queues and change the location but not look at the queue manager for NTPGC3
- Display the queue and location against all other queue managers

but they would not get the option to look at the MQ error log files.

32.1. Object Tailoring Keywords

location_create	Allows the user to create new locations
location_change	Allows the user to update this location
location_display	Allows the user to display this location
location_delete	Allows the user to delete this location
queue_display	Allows the user to display queues and queue lists
msg_create	Allows the user to use the put message dialog
msg_copy	Allows the user to copy messages from one queue to another
msg_move	Allows the user to move messages from one queue to another
msg_display	Allows the user to browse queues
msg_delete	Allows the user to delete messages from queues
qmgr_display	Allows the user to display the Queue Manager definition
location_all	Allows the user to do anything to the location objects
msg_all	Allows the user to do anything to messages
queue_all	Allows the user to do anything to queue objects
qmgr_all	Allows the user to do anything to the Queue Manager definition
all_create	Allows the user to create objects of all types
all_change	Allows the user to change objects of all types
all_display	Allows the user to display objects of all types
all_delete	Allows the user to delete objects of all types
all	Allows the user to do anything to all object types.
queue_list	By default MQEdit will display all the queues that the user has authority to see. Effectively MQEdit will issue a *DISPLAY QUEUE(*)* command. However, by using **queue_list** you can change the command that is sent and reduce the number oof returned queues. For example: **queue_list = "APPGRP1.*"** This will change the command to *DISPLAY QUEUE(APPGRP1*).* Note that only a single name can be provided. If you wish to have more extensive filtering of the names shown then you can also use **queue_mask** below. Clearly the set of names returned by **queue_list** must include all the names that you wish to be able to see.

queue_mask	By default MQEdit will display all the queues that the user has authority to see. However, this option allows you to limit which queues are shown. Filtering is done in MQEdit (not on the server) – see queue_list above. It can be useful to reduce the list of queues presented to the user. The format of the entry should be: **queue_mask = { <Mask1> , <Mask2>, ….}** The list can either be comma or space separated. If the mask starts with a '!' character it indicates that Queues matching the mask should not be shown. The order of the masks is relevant and matching is done in the order of the definition. For example, you can define a set which is shown and then remove a subset. Alternatively you can define that nothing should be displayed and then define a subset which should be displayed. For example: **queue_mask = { *PRD*,!SYS, SYSTEM.DEF* }** This example says display any queues which contain the string 'PRD' except queues which start with 'SYS' should not be shown, except queues which start with 'SYSTEM.DEF' should be shown.
queue_prefix_min	This configuration allows you to specify a minimum number of characters which must be specified in the Queues Prefix option. This can be useful if you want to prevent your users from asking for 'everything' but instead ask for, perhaps, just their application group queues. This can reduce load on the server and reduce the number of authority events which might be generated as a result of asking for all queues. If a non-zero value is specified in the queue_prefix_min field then the Queues Prefix entry field will be shown by default.

32.2. Menu Tailoring Keywords

32.2.1. File Menu

nomenu_about	Removes the named option from the menu
nomenu_add_location	“ “ “
nomenu_copy_location	“ “ “
nomenu_delete_location	“ “ “
nomenu_help	“ “ “
nomenu_import_location	“ “ “
nomenu_open_location	“ “ “
nomenu_preferences	“ “ “
nomenu_refresh_information	“ “ “
nomenu_save_configuration	“ “ “

32.2.2. Edit Menu

nomenu_expert_mode	“ “ “

32.2.3. Action Menu

nomenu_connect_all	“ “ “
nomenu_disconnect	“ “ “
nomenu_disconnect_all	“ “ “
nomenu_filters	“ “ “
nomenu_ini_file	“ “ “
nomenu_log_file	“ “ “
nomenu_publish_message	“ “ “
nomenu_put_message	“ “ “
nomenu_qload	“ “ “

32.2.4. View Menu

nomenu_colours	Removes the named option from the menu
nomenu_font	“ “ “
nomenu_mq_version	“ “ “
nomenu_search	“ “ “
nomenu_view	“ “ “

32.3. Dialog Menus

nomenu_alterlist	Removes the named option from the menu
nomenu_autorefresh	“ “ “
nomenu_context	Removes any context menus (those displayed when button 2 is pressed) from the application.
nomenu_defaultfilter	Removes the named option from the menu
nomenu_export	“ “ “
nomenu_listtitles	“ “ “
nomenu_msg_format	Removes all the menus concerned with message formatting from the message browsing dialogs.

nomenu_msg_ops	Removes all the operations such as Copy, Move, Delete from the message browsing dialogs.
nomenu_msg_selection	Removes all the menus concerned with message selection from the message browsing dialogs.
nomenu_showhide	Removes all the show/hide menu menu options such as 'Show Buttons'
nomenu_splitlist	Removes the named option from the menu

Chapter 33. Trouble Shooting

Because of the nature of messaging the application can not guarantee that replies from remote Queue Managers will be returned. If you find that a command window continually waits for a response the most common reasons are:-

1. The Command server for the Queue Manager is not running
2. The channel to the remote Queue Manager is not running
3. The channel from the remote Queue Manager is not running
4. The user does not have security access to the Queue Manager

   The usual symptom of this is that messages will build up on the remote Queue Manager’s Dead Letter Queue. The 'Q' program (SupportPac MA01) can be used to browse the messages on the Dead Letter Queue to see the reason code.

   The most common cause of this is that the userid in the incoming message does not have a security profile on the receiving machine. For Windows it will be the logged on userid. Therefore, the receiving Queue Manager, be it OS/400, z/OS, AIX or whatever, must have a security profile for this id. Alternatively, Channel exits can be used to change the userid as the message is being transferred.

33.1. Service

As with any software product of any complexity it always possible that there are bugs in the code. If you notice strange behaviour or the program crashes then by all means raise a bug report by sending an email to support@mqgem.com. Before you do this though please read the relevant part of this manual. Most of the 'problems' reported are user errors and have arisen through unfamiliarity of that part of the product. The next thing to ensure is that you are using the latest version of the software. The latest version is always available for download here http://www.mqgem.com/mqedit_download.html Provided you have a current licence you can upgrade to the latest version of the software free of charge.

If upgrading doesn't help and you are certain it is a product bug then please do send an email to support@mqgem.com Please be as specific as possible about the problem you are having and how the problem can be recreated. Screen-shots of the problem often help enormously. Remember that the more detail you put in to your problem description then the faster your problem can be resolved.

33.1.1. Mini-Dump

If the problem is a program crash then it is possible to get MQEdit to generate what is known as a mini-dump. A mini-dump is a small file created at the time of the crash which shows service what the program was doing at the time of the failure. To get MQEdit to catch exceptions and generate a dump file you need to start MQEdit with a '-E' parameter. For example:

```
mqedit -E c:\mqedit_dumps
```

The -E parameter is followed by the directory where you would like the dump files written. The files will have a name something like **mqedit_20141029_092103_114692.dmp**. The first parts of the file name are the name of the program, the date and timestamp. There are actually two types of mini-dump. The default mini-dump file contains only stack storage but is good enough to solve many problems since it shows where the program was and what it was doing at the time of the dump. However, it doesn't contained any heap storage which is why the file is only around 50K. There are times though where a dump file containing the heap storage is needed. The files are often typically 100MB in size. To obtain a full dump file you need to add the parameter **-F full** to MQEdit. With this parameter all dump files generated, by any means, will also contain the heap storage. Please do not send full dumps by email.

At any time, even if you don't use the -E parameter, you can ask MQEdit to generate a dump file by pressing <ALT>,<SHIFT> and '1' keys all at the same time. If the -E parameter is not specified the file will be written to the same directory as the program.

Generally speaking you would create a dump file only when directed by service. However, if you have a crash then by all means send the dump file along with your problem report. Remember though that a dump file is not a replacement for good problem text. It can show what the program is doing at that instant but not how it got there. It is just a tool which **can** help. Always include the program version and build date from the About box.

Chapter 34. Changes made in previous versions

Changes made in Version 9.3.2

The following changes have been made in this version:

1. **MQEdit converted to a 64-bit program**
   IBM MQ have announced that they intend to drop support for 32-bit applications. While they haven't said exactly when this will happen they have indicated that it may be fairly soon. So, the sole reason for this release is to convert MQEdit to a 64-bit program and ensure that it is runnable in all environments. Clearly this means that there is a migration step to ensure that MQEdit now has access to the 64-bit rather than 32-bit IBM MQ product libraries.

   Of course anyone who has ever ported an application, of any complexity, from 32-bit to 64-bit will know that it is a lot more involved than just recompiling with a new set of options. There can be lots of unintended consequences that are not easily predictable. For this reason we have kept other changes to a minimum. There are very few actual functional changes in MQEdit 9.3.2 compared with MQEdit 9.3.1. The focus is to ensure that the program operates, at least externally, exactly like it's predecessor.

Changes made in Version 9.3.1

1. **KDB Password prompting**
   Have MQEdit prompt you for the password to open the Key Database file when it is needed rather than storing it anywhere. For more details see.22.4. Key Database File Password on page 126.

2. **Separation of context menu for Add and Update operations**
   Previous versions of MQEdit had a single menu setting for type of context which would be used for the message edit operation. Version 9.3.1 adds the ability to set a different context to be used for an 'Add' or 'Update' operation.

   For more information please see Section 3.9 Message Context on page 21.

Changes made in Version 9.3.0

1. **Ability to specify a queue prefix so only a portion of the queues are returned**
   For more information please see Section 18.4.3 Queues Prefix on page 102.

2. **Preference option to control whether INQUIRE QUEUE or INQUIRE QUEUE NAMES command is used**
   For a description of this please see Section 19.1.4 Use INQUIRE QUEUE NAMES on page 105.

Changes made in Version 9.2.1

1. **Better integration with messages produced and consumed by COBOL**
   With the introduction of the free MakeFmt utility it is now far easier and faster to generate MQEdit user format files from COBOL copybooks. This makes displaying and editing MQ messages built by COBOL programs a breeze. See the MakeFmt User Guide (available at https://www.mqgem.com/makefmt.html) for more information.

2. **Ability to specify an Override section in the User Format definition**
   Please see 'Override Section' on page 87 for more information.

3. **Support for Packed Decimal fields**
   Support for this field type makes display and editing of COBOL copybooks much easier. Please see 'Packed Decimal Fields' on page 78 for more information.

4. **Allow arrays of character arrays**
   Please see 'Fixed Arrays' on page 81 for more information.

5. **Support for s/390 FLOAT and DOUBLE field editing**
   For more information about floating point fields in MQEdit please see 'Floating Point Numbers' on page 88.

6. **Addition of field description values in User Formats**
   Please see 'Field Description' on page 79 for more information.

7. **Addition of character based enumeration definitions**
   These enumerations allow you to specify 'known' values for character, packed, float and double fields. For more information please see 'Character Enumeration' on page 85.

8. **Show enumeration and field description values in message summary**
   When using 'User Format Messages' you can identify any combination of fields to be shown in the Message Summary. If those fields have an associated 'Field Description' or a 'Field Description' then that value will now be shown in the Message Summary.

9. **IBM MQ Command Level 924 Support**

10. **Allow description to be added after user format definition**

Changes made in Version 9.2.0

1. **Ability to make multiple changes to your locations in a single click**
   To make it easier to configure and organise your locations you can now select a group of locations and change a location settings, say the Network Names field, in one go. For more information please see Chapter 21 Multiple Location Change Dialog on page 120.

2. **Addition of the concept of Security Groups**
   It is now possible to identify groups of locations as belonging to the same security 'group'. This means that each member of the group will use the same logon userid/password. For more information, please see “Security Group“ description on page 117.

3. **New options to automatically fold Userid and Password fields to uppercase**
   Please see “Upper Case Userid / Password“ on page 117 for further information.

4. **Main window search fields remember previously used search strings**
   The two search fields are now combo-boxes. Select from the drop-down to pick previously used search strings.

Changes made in Version 9.1.0

1. **Character Substitution Mode**
   To make is easier to read certain types of messages MQEdit now supports a Character Substitution Display Mode. This will display text such as **&gt;** as the character '>'. For more information please see 'Character Substitution Mode' on page 37.

2. **QLOAD 'other' file processing**
   It is now possible to use the QLOAD feature to load a 'normal' text file rather than a standard QLOAD file. For more information, please see 'Other File ' on page 94.

3. **Select All**
   To make it easier to copy/paste the entire message a new Select All (Ctrl-A) action has been added.

Changes made in Version 9.0.3

1. **Master Configuration File**
   MQEdit now has the notion of a master configuration file. This is passed as a parameter on start of the program and causes any locations definitions to be copied from that configuration file in the current configuration. Please see Chapter 26.3 Master Configuration File on page 132 for more information.

2. **Import from another MQEdit Configuration File**
   MQedit now supports migrating location definition from another MQEdit configuration file. Please see 18.1.6 Import Locations on page 98 for more information.

3. **Licence Command**
   MQEdit can now be configured to run an OS command when the licence file is either not present or has less than 30 days to run. For more information please see Chapter 26.4 Licence Command on page 132.
4. **Splash Screens**
   It is now possible to configure MQEdit to display a splash screen upon start-up and when connecting to a Queue Manager for the first time. For more information please see Chapter 26.2 Splash Screens on page 130.
5. **MQEdit Program update notification**
   It is always recommended that you run the latest version of MQEdit and get the benefit of new features. To make this easier MQEdit will now check whether there is a new version and display a dialog to you.
6. **Support of CCDT URL**
   MQ V9 allows a connecting application to specify the URL location of the CCDT file to use. This field can be specified in the location dialog.
7. **Improved performance on machines with large numbers of persistent dynamic queues**
8. **Added ability to show the number of queues available next to each location entry**
9. **Added ability to store and restore default Queue and Location search strings**

Changes made in Version 9.0.2

1. **JSON message support**
2. **FIX message support**
   FIX stands for Financial Information eXchange and is widely used in real time exchange of information related to securities transactions and markets.
3. **CSV (Comma Separated Values) message support**
   A simple multi-field message where each field is delimited by a comma. This format is used by OFS (Open Financial Standard) messaging.

Changes made in Version 9.0.0

Initial Version

Appendix A: Over-punched ASCII format

While the format for over-punched numbers is well-defined in EBCDIC there are actually two possible formats that could be used for data that is in ASCII. Which format is used depends on the compiler in question.

Modified (same letters as EBCDIC)

This format uses the same character representation as EBCDIC. Thus if an MQ message was converted from EBCDIC to ASCII the same set of letters would mean the same numeric and signed representation, even though the hexadecimal values would bear no relationship to the numeric value.

Digit	EBCDIC Hex-Binary	EBCDIC Char	ASCII Hex	ASCII Char
`+0`	`X'C0' - 1100 0000`	`{`	`X'7B' - 0111 1011`	`{`
`+1`	`X'C1' - 1100 0001`	`A`	`X'41' - 0100 0001`	`A`
`+2`	`X'C2' - 1100 0010`	`B`	`X'42' - 0100 0010`	`B`
`...`				
`-0`	`X'D0' - 1101 0000`	`}`	`X'7D' - 0111 1101`	`}`
`-1`	`X'D1' - 1101 0001`	`J`	`X'4A' - 0100 1010`	`J`
`-2`	`X'D2' - 1101 0010`	`K`	`X'4B' - 0100 1011`	`K`

PL/I compilers on ASCII systems use this.

Strict

This format uses an alternative hexadecimal value for the negative numbers in ASCII which does bear a relationship to the number. The strict definition for zoned, or over punched numbers is that each digit is represented by a zoned bit value in the left-most four bits (nibble or half-word) and the binary value for the digit in the right-most four bits (nibble or half-word).

Digit	EBCDIC Hex-Binary	EBCDIC Char	ASCII Hex	ASCII Char
`+0`	`X'C0' - 1100 0000`	`{`	`X'30' - 0011 0000`	`0`
`+1`	`X'C1' - 1100 0001`	`A`	`X'31' - 0011 0001`	`1`
`+2`	`X'C2' - 1100 0010`	`B`	`X'32' - 0011 0010`	`2`
`...`				
`-0`	`X'D0' - 1101 0000`	`}`	`X'70' - 0111 0000`	`p`
`-1`	`X'D1' - 1101 0001`	`J`	`X'71' - 0111 0001`	`q`
`-2`	`X'D2' - 1101 0010`	`K`	`X'72' - 0111 0010`	`r`

Gnu COBOL and MicroFocus COBOL use this.

Which format does MQEdit use?

MQEdit uses the modified format as that seems to make the most sense for data in MQ messages. Our thinking is that if you are using these types of fields, it is because keeping all the data as character data is important to you, and MQ will data convert the character data for you into ASCII, so using the same letter representation would be what you'd likely be using. We are interested in feedback on this decision. Do you have over-punched data in ASCII in your messages? Are you using the modified or strict data format? Do you need MQEdit to support both?

End of document

MQGem Software Limited

www.mqgem.com