MQ Message Encryption Installation and Operation Manual

Capitalware Inc.
Unit 11, 1673 Richmond Street, PMB524
London, Ontario N6G2N3
Canada
sales@capitalware.com
https://www.capitalware.com

Last Updated: January 2022.

Table of Contents

1 Introduction

1.1 Overview

MQ Message Encryption (MQME) provides encryption for MQ message data while it resides in a queue or topic and in the MQ logs (i.e. all data at rest). In cryptography, encryption is the process of transforming information into an unreadable form (encrypted data). Decryption is the reverse process. It makes the encrypted information readable again. Only those with the key (PassPhrase) can successfully decrypt the encrypted data. MQME uses Advanced Encryption Standard (AES) to encrypt the data. AES is a data encryption scheme, adopted by the US government, that uses three different key sizes (128-bit, 192-bit, and 256-bit).

One of the features that MQME offers is the ability to control who accesses protected queues/topics. This control is obtained through the use of UserId grouping. MQME can query the local OS group or a group file. Group files are implemented in a similar manner to the way they are implemented in Unix and Linux (i.e. **/etc/group** file). Normally, the 'mqm', 'QMQM' or 'MUSR_MQADMIN' MQ UserIds or any UserId in the 'mqm' group get full access to all messages in all queues/topics. For queues/topics protected by MQME, those privileged UserIds do **not** get access to the messages in the protected queues/topics unless they are explicitly added to the authorized list of users or groups.

Another feature of MQME is its ability to generate and validate the message via a digital signature. MQME uses the SHA-2 to create a cryptographic hash function (digital signature) for the message data. The digital signature provides verification that the message data has not been altered.

MQME is an MQ API Exit that operates with IBM MQ v7.1, v7.5, v8.0, v9.0, v9.1 and v9.2 in Windows, Unix, IBM i (OS/400) and Linux platforms.

On AIX, HP-UX, Linux, Solaris and Windows, MQME can be configured and used with a non-default installation of MQ in a multi-install MQ environment.

Note: Raspberry Pi is a Linux ARM 32-bit OS (Operating System). Hence, simply follow the Linux 32-bit instructions for installing and using the solution on a Raspberry Pi.

1.2 Executive Summary

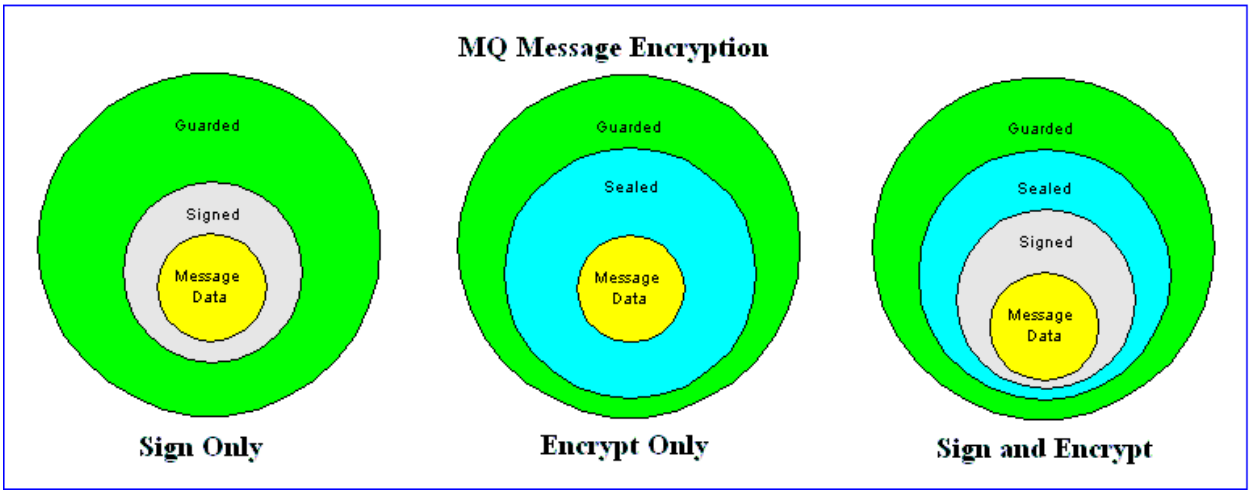
MQME is an MQ API Exit. The MQ API Exit is available in 3 forms:

- Windows DLL
- Shared library for AIX, HP-UX, Linux, and Solaris.
- IBM i exit module

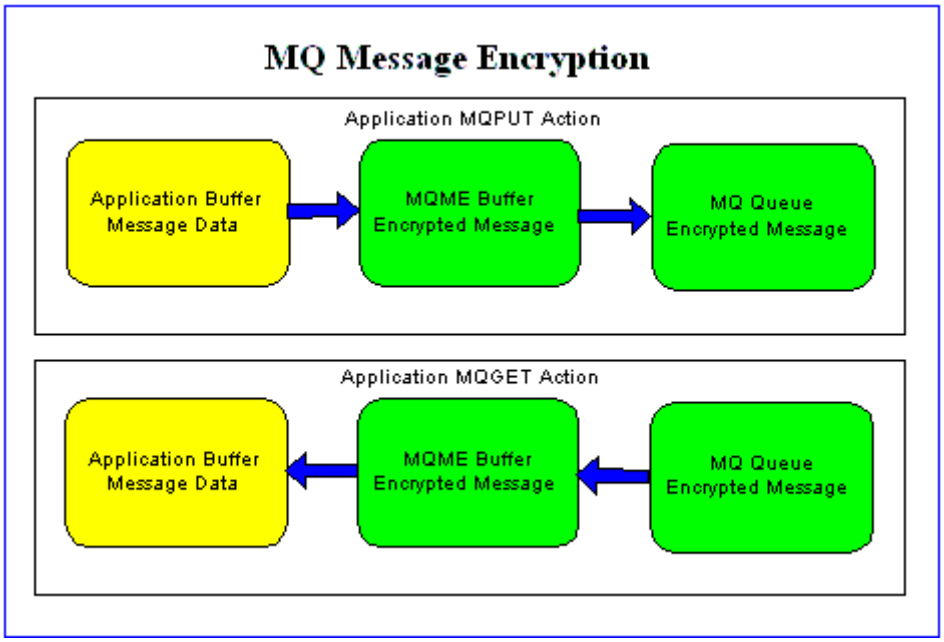
The major features of MQME are as follows:

- No application changes required
- All message data written to a selected queue and/or topic will be encrypted (nothing missed or forgotten)
- Secure encryption/decryption methodology using AES with 128, 192 or 256-bit keys
- Uses the SHA-2 to create a cryptographic hash function (digital signature)
- No application changes required
- Group authority checking against the local OS groups or a group file
- Standard MQ feature, GET-with-Convert, is supported
- Provides high-level logging capability for encryption/decryption processing

1.3 Message Diagram (Logical View)



1.4 Message Flow (Logical View)



1.5 Prerequisites

This section provides the minimum supported software levels. These prerequisites apply to server-side installations of MQ Message Encryption.

1.5.1 Operating System

MQ Message Encryption can be installed on any of the following supported servers:

1.5.1.1 IBM AIX

- IBM AIX 6L version 6.1 or higher

1.5.1.2 HP-UX IA64

- HP-UX v11.23 or higher

1.5.1.3 IBM i (OS/400)

- IBM i V6R1 or higher

1.5.1.4 Linux x86

- Red Hat Enterprise Linux v5, v6, v7, v8
- SUSE Linux Enterprise Server v11, v12, v15

1.5.1.5 Linux x86_64 (64-bit)

- Red Hat Enterprise Linux v5, v6, v7, v8
- SUSE Linux Enterprise Server v11, v12, v15

1.5.1.6 Linux on POWER

- Red Hat Enterprise Linux v5, v6, v7, v8
- SUSE Linux Enterprise Server v11, v12, v15

1.5.1.7 Linux on zSeries (64-bit)

- Red Hat Enterprise Linux v5, v6, v7, v8
- SUSE Linux Enterprise Server v11, v12, v15

1.5.1.8 Raspberry Pi (Linux ARM 32-bit)

- Raspberry Pi OS v9 or higher

1.5.1.9 Sun Solaris

- Solaris SPARC v10 & v11
- Solaris x86_64 v10 & v11

1.5.1.10 Windows

- Windows 2008, 2012 or 2016 Server (32-bit & 64-bit)
- Windows 7, 8, 8.1 or 10 (32-bit & 64-bit)

1.5.2 IBM MQ

- IBM MQ v7.1, v7.5, v8.0, v9.0, v9.1 and v9.2 (32-bit and 64-bit)

Operating System	MQ v7.1, v7.5, v8.0, v9.0, v9.1 and v9.2
AIX v6.1 or higher	32-bit & 64-bit
HP-UX IA64 v11.23 or higher	32-bit & 64-bit
IBM i (OS/400)	64-bit
Linux x86	32-bit
Linux x86_64	32-bit & 64-bit
Linux on POWER	32-bit & 64-bit
Linux on zSeries	32-bit & 64-bit
Raspberry Pi ARM	32-bit
Solaris SPARC v10 & v11	32-bit & 64-bit
Solaris x86_64 v10 & v11	32-bit & 64-bit
Windows 2008, 2012, 2016, 7, 8, 8.1 & 10	32-bit & 64-bit

1.5.3 Windows 32-bit

The following is the software prerequisite for Windows 32-bit:

- Microsoft Visual C++ 2010 Redistributable Package (x86)
  https://download.microsoft.com/download/1/6/5/165255E7-1014-4D0A-B094-B6A430A6BFFC/vcredist_x86.exe

1.5.4 Windows 64-bit

The following are the software prerequisite for Windows 64-bit:

- Microsoft Visual C++ 2010 Redistributable Package (x64)
  https://download.microsoft.com/download/1/6/5/165255E7-1014-4D0A-B094-B6A430A6BFFC/vcredist_x64.exe

If local 32-bit applications connect in bindings mode to the queue manager then the following needs to be also installed:

- Microsoft Visual C++ 2010 Redistributable Package (x86)
  https://download.microsoft.com/download/1/6/5/165255E7-1014-4D0A-B094-B6A430A6BFFC/vcredist_x86.exe

2 Installing MQ Message Encryption

This section describes how to install Capitalware's MQME.

2.1 API Exit

2.1.1 Windows Installation

To install MQME on Windows, first unzip the **mqme.zip** and then run the **mqme_setup.exe** file. Follow the on-screen instructions and the API exit will be installed in the **C:\Capitalware\MQME** directory (default installation).

The user may copy or ftp the mqme.dll, 64\mqme.dll, rotatelog.bat and mqme.ini files from one Windows server to another Windows server.

2.1.2 Linux 32-bit Installation

To install the 32-bit version of MQME on Linux, first unzip the **mqme.zip** and then select the appropriate TAR file for the target platform. You will find 2 TAR file in the original ZIP file:

- **Linux_x86/mqme_linux.tar**
- **RaspberryPi_ARM/mqme_raspberrypi_arm.tar**

Steps to Install MQME:

- ftp or copy the selected TAR file to the target platform to the ***/var/mqm/*** directory.
- Un-tar the **mqme_xxx.tar** file into the ***/var/mqm/*** sub-directory (xxx is either aix, hpux, solaris or linux)

```
cd /var/mqm/
tar -xvf  mqme_xxx.tar
```

- Change directory to ***/var/mqm/exits/***
- Next, do the following commands against ***mqme***:

```
chmod +x setmqme.sh
./setmqme.sh
```

Note: It is important that both the License file and the IniFile are world-readable. Issue the following commands to change the file permissions:

```
cd /var/mqm/exits/
chmod 644 mqme_licenses.ini  mqme.ini
```

2.1.3 Unix and Linux 64-bit Installation

To install the 64-bit version of MQME on Unix or Linux, first unzip the **mqme.zip** and then select the appropriate TAR file for the target platform. You will find 7 TAR files in the original ZIP file:

- **AIX/mqme_aix71_64.tar** for AIX v7.1 or higher
- **HPUX_IA64 /mqme_hpux64_ia64.tar**
- **Linux_x86_64/mqme_linux_x86_64.tar**
- **Linux_POWER/mqme_linux_power64.tar**
- **Linux_zSeries/mqme_linux_zseries64.tar**
- **Solaris_SPARC/mqme_solaris10_64.tar** for Solaris SPARC v10 or higher
- **Solaris_x86_64/mqme_solaris_x86_64.tar**

Steps to Install MQME:

- ftp or copy the selected TAR file to the target platform to the ***/var/mqm/*** directory.
- Un-tar the **mqme_xxx.tar** file into the ***/var/mqm/*** sub-directory (xxx is either aix, hpux, solaris or linux)

```
cd /var/mqm/
tar -xvf  mqme_xxx64.tar
```

- Change directory to ***/var/mqm/exits64/***
- Next, do the following commands against ***mqme***:

```
chmod +x setmqme.sh
./setmqme.sh
```

Note: mqme_r shared library is not required for Solaris.

Note: It is important that both the License file and the IniFile are world-readable. Issue the following commands to change the file permissions:

```
cd /var/mqm/exits64/
chmod 644 mqme_licenses.ini  mqme.ini
```

2.1.4 IBM i Installation

To install the MQME on IBM i, first unzip the **mqme.zip** and then select the files in the IBM i (iSeries) directory.

- **mqme.savf** is the IBM i 'Save File' that contains the library with the API exit.
- **mqme_iseries.tar** is the IBM i IFS TAR file that contains a sample initialization file for the API Exit.

Steps to install the API Exit:

1. Log onto the target IBM i server and do the following command:

   ```
   CRTSAVF FILE(QGPL/MQME)
   ```

2. ftp the IBM i files to the IBM i server as follows:

   ```
   ftp -s:mqme_iseries.ftp  iseries_hostname
   ```

   The mqme_iseries.ftp file contains the following ftp commands:

   ```
   your-IBM i-userid
   your-IBM i-password

   binary
   cd QGPL
   put mqme.savf

   quote SITE NAMEFMT 1

   cd /QIBM/UserData/mqm/
   put mqme_iseries.tar
   quit
   ```

3. Log onto the target IBM i server and do the following commands:

   ```
   RSTLIB SAVLIB(MQME) DEV(*SAVF) SAVF(QGPL/MQME)
   CLRSAVF FILE(QGPL/MQME)
   CHGOBJOWN  OBJ(MQME) OBJTYPE(*LIB)  NEWOWN(QMQM)
   qsh
   cd /QIBM/UserData/mqm/
   tar -xvf mqme_iseries.tar
   chown -R QMQM mqme
   chmod -R 777 mqme
   rm mqme_iseries.tar
   ```

2.1.5 MQME-GUI Installation

This section will describe how to install the MQME-GUI. The user will find 2 files in the software package listed as follows:

- **MQME-GUI/mqmegui-wthjre.exe** (for Windows)
- **MQME-GUI/mqmegui.zip** (for Unix, Linux or macOS)

2.1.5.1 MQME-GUI Installation on Windows

To install MQME-GUI on Windows, run the **mqmegui-withjre.exe** file located in the MQME-GUI directory. Follow the on-screen instructions and the program will be installed in the **C:\ Capitalware\MQME-GUI** directory (default installation).

2.1.5.2 MQME-GUI Installation on Unix, Linux or macOS

To install MQME-GUI on Unix or Linux, you will need to ftp or copy the selected TAR file to the target platform to the /home/mqm/ directory. Next, one must telnet to the Unix, Linux or macOS server and 'cd' (change directory) to the /home/mqm/ directory and unzip the archive file.

i.e. Do the following command:

```
unzip mqmegui.zip
```

3 Configuring MQME

This section describes how to configure the MQME (MQ API Exit) on the support platforms.

Platform		Directory	Exit Module Name
Windows	32-bit	C:\Capitalware\MQME\	mqme.dll
Windows	64-bit	C:\Capitalware\MQME\64\	mqme.dll
Linux/Unix	32-bit	/var/mqm/exits/	mqme
Linux/Unix	64-bit	/var/mqm/exits64/	mqme

MQME supports MQ's multi-install in a non-default directory.

After the user has configured the MQME, the queue manager needs to be restarted.

3.1 API Exit

This section describes the necessary entries to enable the API Exit. The MQ Administrator will need to create a Local API Exit definition. This procedure differs depending on the platform.

Note: The value for the Data parameter cannot exceed a length of 32 characters.

3.1.1 Windows

On Windows, there are 2 ways to create the Local API Exit definition: via the command line or MQ Explorer. Note: Using MQ Explorer to add the API Exit is very easy.

3.1.1.1 Windows Command Line

First, the user will need to manually edit the queue manager's qm.ini file to update the ExitsDefaultPath and ExitsDefaultPath64 fields. The qm.ini file will be located at: ***C:\Program Files (x86)\IBM\IBM MQ\Qmgrs\{QMgrName}\qm.ini***

```
ExitPath:
   ExitsDefaultPath=C:\Capitalware\MQME;C:\Program Files (x86)\IBM\IBM MQ\exits
   ExitsDefaultPath64=C:\Capitalware\MQME\64;C:\Program Files (x86)\IBM\IBM MQ\exits64
```

Next, create the Local API definition via the command line using the MQ ***amqmdain*** program. The MQAdmin can issue the following commands to create the Local API Exit definition:

```
amqmdain reg {QMgrName} -c add  -s ApiExitLocal\MQME -v Name=MQME
amqmdain reg {QMgrName} -c add  -s ApiExitLocal\MQME -v Module=mqme.dll
amqmdain reg {QMgrName} -c add  -s ApiExitLocal\MQME -v Data=C:\Capitalware\MQME\mqme.ini
amqmdain reg {QMgrName} -c add  -s ApiExitLocal\MQME -v Sequence=2
amqmdain reg {QMgrName} -c add  -s ApiExitLocal\MQME -v Function=EntryPoint
```

Where ***{QMgrName}*** is the name of the Queue Manager.

As a convenience, we have included a batch file called ***mqme_reg.bat*** that includes all of the **amqmdain** commands. **mqme_reg.bat** is located in the MQME install directory and it accepts one parameter (QMgrName).

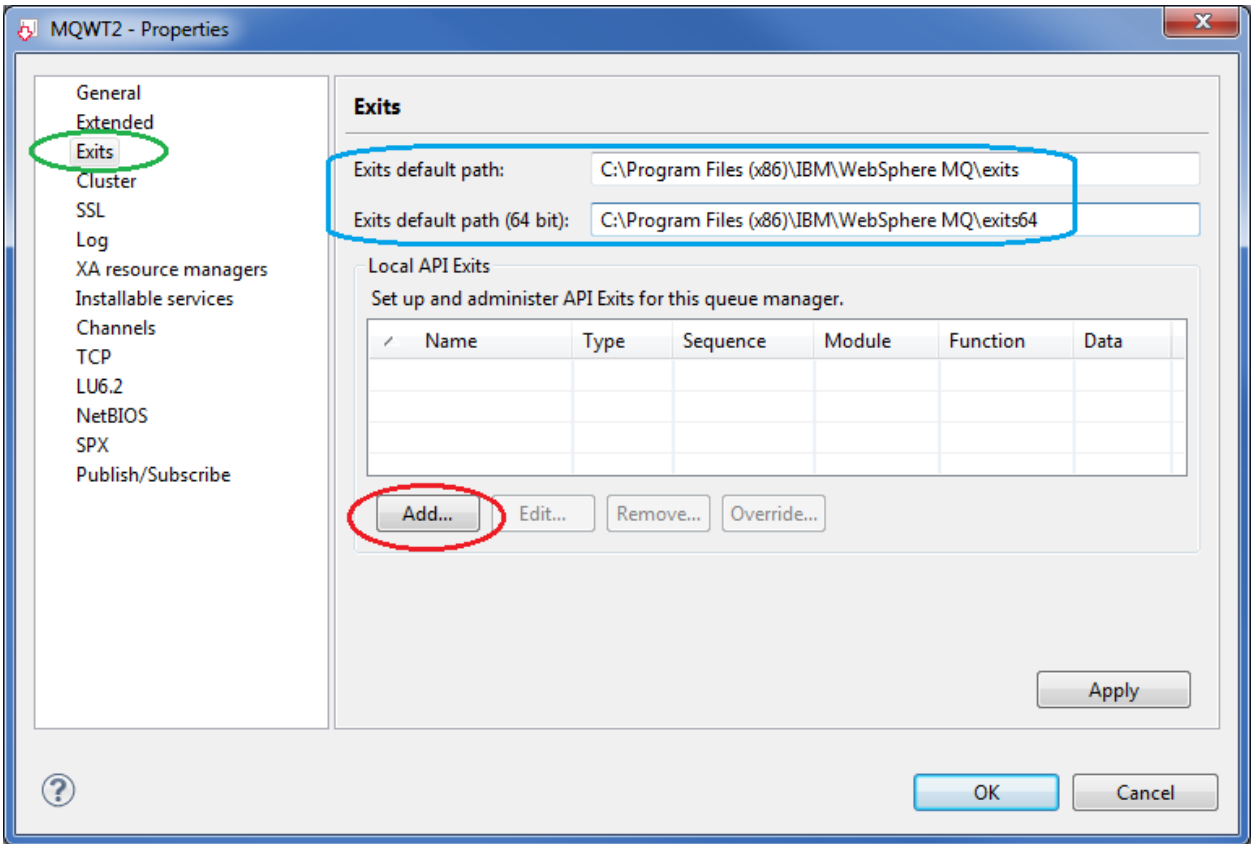
Open a Windows Command Prompt and issue the following commands to configure MQME for a particular queue manager:

```
cd /D C:\Capitalware\MQME\
mqme_reg.bat {QMgrName}
```

3.1.1.2 Windows MQ Explorer

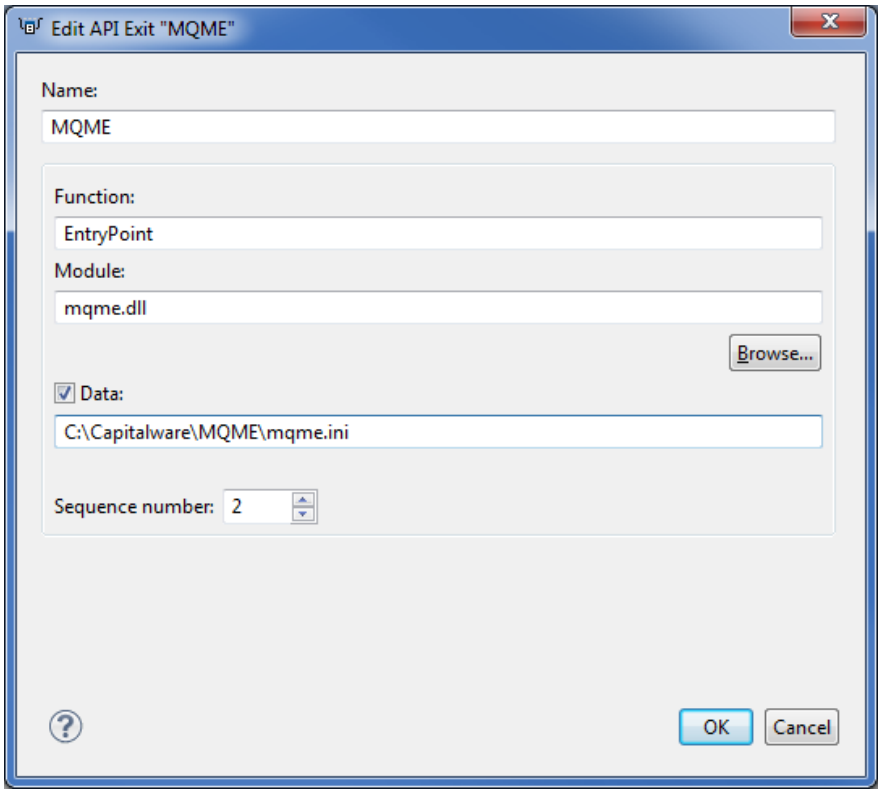
To create a Local API Exit definition using MQ Explorer, do the following:

- Start MQ Explorer
- Right-click on the queue manager name
- Select ***Properties*** from the popup menu
- Select ***Exits*** in the left panel of the properties window
- Update the 'Exit default path' and 'Exit default path (64-bit)' with the MQME install path. i.e. C:\Capitalware\MQME and C:\Capitalware\MQME\64
- Click the ***Add*** button.



The user is required to input the following values into the 5 fields of the Add API Exit window:

Field Name	Value
Name	MQME
Function	EntryPoint
Module	mqme.dll
Data	C:\Capitalware\MQME\mqme.ini
Sequence Number	2



Click ***OK*** when the information has been inputted.

3.1.2 For Linux 32-bit

Edit the ***qm.ini*** for the queue manager that MQME is being applied to. The **qm.ini** file is located at **/var/mqm/qmgrs/{QMgrName}/qm.ini**

Make sure the ExitPath stanza exists in the qm.ini and then add the ApiExitLocal stanza as given below:

```
ExitPath:
   ExitsDefaultPath=/var/mqm/exits/

ApiExitLocal:
  Name=MQME
  Sequence=2
  Function=EntryPoint
  Module=mqme
  Data=/var/mqm/exits/mqme.ini
```

Note: If the user has not installed MQME in /var/mqm/exits/ directory then the ExitsDefaultPath field will need to be updated with the path to the shared library.

3.1.3 Unix and Linux 64-bit

Edit the ***qm.ini*** for the queue manager that MQME is being applied to. The **qm.ini** file is located at **/var/mqm/qmgrs/{QMgrName}/qm.ini**

Make sure the ExitPath stanza exists in the qm.ini and then add the ApiExitLocal stanza as given below:

```
ExitPath:
   ExitsDefaultPath=/var/mqm/exits/
   ExitsDefaultPath64=/var/mqm/exits64/

ApiExitLocal:
  Name=MQME
  Sequence=2
  Function=EntryPoint
  Module=mqme
  Data=/var/mqm/exits64/mqme.ini
```

Note: If the user has not installed MQME in /var/mqm/exits/ and /var/mqm/exits64/ directories then the ExitsDefaultPath and ExitsDefaultPath64 fields will need to be updated with the path to the shared library.

3.1.4 IBM i

On IBM i, edit the ***qm.ini*** for the queue manager that MQME is being applied to. The qm.ini file is located at **/QIBM/UserData/mqm/qmgrs/{QMgrName}/qm.ini**

Add the ApiExitLocal stanza as given below:

```
ApiExitLocal:
  Name=MQME
  Sequence=2
  Function=EntryPoint
  Module=MQME/MQME
  Data=/QIBM/UserData/mqm/mqme/mqme.ini
```

3.2 File Paths

Data field of the ApiExitLocal stanza must NOT exceed 32 characters. In order to work with this limitation, MQME supports 3 ways to specify an IniFile path: absolute path, relative path and environment variable.

Note: The IniFile path that is determined by MQME API Exit will also be used for the following IniFile keywords (if no pathing is specified for these keywords): **LicenseFile** and **LogFile**.

3.2.1 Absolute Path

Absolute pathing (specifying the complete path) for the Data field works on Linux, Unix and Windows platforms.

E.g. Windows

```
Data=C:\Capitalware\MQME\mqme.ini
```

Hence, MQME will use the following path as the IniFile path:
C:\Capitalware\MQME

3.2.2 Relative Path

Relative pathing for the Data field is supported on Linux, IBM i, Unix and Windows platforms. MQME will extract the path from Data field and prefix it to the IniFile specified in the Data field in order to locate the IniFile.

For Windows:
C:\Capitalware\MQME

For IBM MQ 32-bit on Unix and Linux:
/var/mqm/exits/

For IBM MQ 64-bit on Unix and Linux:
/var/mqm/exits64/

For IBM MQ on IBM i:
/QIBM/UserData/mqm/mqme/

E.g. Unix

```
Data=mqme.ini
```

Hence, MQME will use the following path as the IniFile path:
E.g. 32-bit
/var/mqm/exits/
E.g. 64-bit
/var/mqm/exits64/

3.2.3 Environment Variables

3.2.3.1 Global Environment Variable

MQME supports the use of the MQME_HOME environment variable which holds the directory path information. MQME_HOME environment variable is supported on Linux, IBM i, Unix and Windows platforms.

e.g. Unix

```
export MQME_HOME=/really/long/path/MQHA/QMgrName/data/
```

```
Data=mqme.ini
```

Hence, MQME will use the following path as the IniFile path:
/really/long/path/MQHA/QMgrName/data/

3.2.3.2 Queue Manager Specific Environment Variable

MQME supports the use of the MQME_HOME environment variable post-fixed with the queue manager name which holds the directory path information. MQME_HOME environment variable post-fixed with the queue manager name is supported on Linux, IBM i, Unix and Windows platforms.

e.g. Unix with a queue manager name of MQA1

```
export MQME_HOME_MQA1=/really/long/path/MQHA/QMgrName/data2/
```

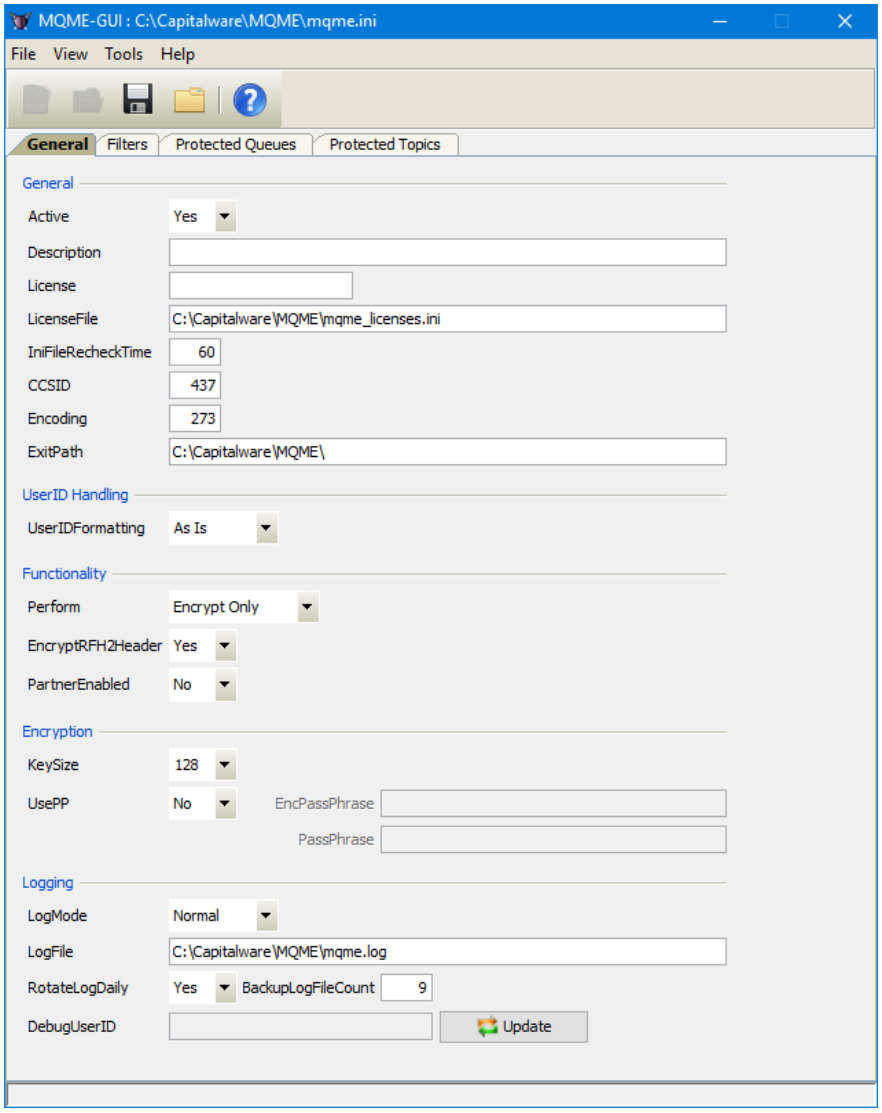
```
Data=mqme.ini
```

Hence, MQME will use the following path as the IniFile path:
/really/long/path/MQHA/QMgrName/data2/

Note: If both environment variables are specified then the queue manager specific environment variable will be used.

3.3 MQME-GUI

This section briefly describes the new graphical program called MQME-GUI. This program assists the user in creating and managing their MQME IniFiles. For more information, please see the ***MQME-GUI User Guide*** manual.



4 IniFile Keywords (Global Values)

This section describes IniFile keywords.

4.1 Active

MQME can be enabled or disabled by the IniFile's property keyword 'Active'. When Active has a value of Y, MQME is activated. The default value is Y.

```
Active=Y
```

4.2 UserIDFormatting

This section describes the necessary entries on how to handle the incoming UserID. 'UserIDFormatting' supports 3 values [A / U / L]. ('As Is, Uppercase and Lowercase). The default value is A.

```
UserIDFormatting=U
```

4.3 KeySize

KeySize specifies a global AES key size used for the encryption/decryption of the message data. Valid values are 128, 192 or 256. The default value is 128.

```
KeySize=128
```

4.4 Perform

Perform indicates which functionality that MQME will perform. Perform supports 3 values [S / E / B]. The default value is E.

- **S** means that MQME will only sign the message
- **E** means that MQME will only encrypt the message
- **B** means that MQME will sign and encrypt the message

When signing the message, MQME creates the digital signature using cryptographic hash function of SHA-2.

```
Perform=E
```

4.5 EncPassPhrase, PassPhrase and UsePP

To enable the use of the user's own PassPhrase, you need 2 keywords in the IniFile:

- **UsePP** allows the use of a user specified PassPhrase
- **EncPassPhrase** specifies a global encrypted PassPhrase for this IniFile that will be used for the protected queues/topics. See Appendix C for details on creating the encrypted PassPhrase.
- **PassPhrase** specifies the actual PassPhrase that will be used for the message encryption and/or decryption (can be 16, 24 or 32 characters/digits in length).

What not to use for your PassPhrase:

- A famous quotation from literature, holy books, etc.
- Something easily guessed by intuition

What to use for your PassPhrase:

- A random selection of characters and numbers
- Use a mix of upper and lower characters
- Use special characters like slash, dot, comma, ampersand, etc.

Encrypted PassPhrase (See Appendix C for details on creating the encrypted passphrase.):

```
UsePP=Y
EncPassPhrase=jXzFNIKKwZ52wsQ3CUwqWUBpDaoVRDnLMDkNqhVEOcswMA
```

Plain text PassPhrase:

```
UsePP=Y
PassPhrase=AeKWU31_wky6MZrL
```

Note: If EncPassPhrase keyword is specified then the PassPhrase keyword is ignored.

4.6 ExcludeApplications

This section describes how to explicitly exclude applications from the encryption/decryption processing when an application has connected to the queue manager. MQME will look up the regular expression patterns from the **ExcludeApplications** keyword in order to determine if the application name matches any of the specified expression patterns.

If there is a match, the application is excluded from encryption/decryption process. Each regular expression pattern is separated from the next pattern by a semi-colon (';').

In the regular expression pattern:

- '*' matches any sequence of characters (zero or more)
- '?' matches any single character
- '#' matches any single numeric digit (0-9)
- '@' matches any single alphabetic character (A-Z, a-z)
- [SET] matches any of the characters in the specified set
- [!SET] or [^SET] matches any character except those specified in the set (negation).

A SET can be composed of characters or ranges. A range is in the form: 'character – character' (i.e. 0-9 or A-Z). Although this is the simplest range allowed in the [ ] pattern, more complex inclusive ranges such as [0-9a-zA-Z] are allowed. [0-9a-zA-Z] specifies that the character can be 0 through 9 **or** a through z **or** A through Z. Other characters are allowed (ie. 8 bit characters) if your system supports them.

In order to suppress the special syntactic significance of any of these characters '[] * ? # @ ! ^ - \', a backslash ('\') must precede the special character.

Note: ExcludeApplications must NOT exceed 2048 characters.

To authorize MQME decryption by application name, you need 2 keywords in the IniFile:

- **UseExcludeApplications** activates this feature. If it has a value of Y, MQME will exclude applications from the encryption/decryption process.
- **ExcludeApplications** specifies the list of application names that are to be excluded.

```
UseExcludeApplications=Y
ExcludeApplications=abc*;gadget;xyz*
```

4.7 ExcludeQueues

This section describes how to explicitly exclude queues from the encryption/decryption processing when an application has connected to the queue manager. MQME will look up the regular expression patterns from the **ExcludeQueues** keyword in order to determine if the queue name matches any of the specified expression patterns.

If there is a match, the queue is excluded from encryption/decryption process. Each regular expression pattern is separated from the next pattern by a semi-colon (';').

In the regular expression pattern:

- '*' matches any sequence of characters (zero or more)
- '?' matches any single character
- '#' matches any single numeric digit (0-9)
- '@' matches any single alphabetic character (A-Z, a-z)
- [SET] matches any of the characters in the specified set
- [!SET] or [^SET] matches any character except those specified in the set (negation).

A SET can be composed of characters or ranges. A range is in the form: 'character – character' (i.e. 0-9 or A-Z). Although this is the simplest range allowed in the [ ] pattern, more complex inclusive ranges such as [0-9a-zA-Z] are allowed. [0-9a-zA-Z] specifies that the character can be 0 through 9 **or** a through z **or** A through Z. Other characters are allowed (ie. 8 bit characters) if your system supports them.

In order to suppress the special syntactic significance of any of these characters '[] * ? # @ ! ^ - \', a backslash ('\') must precede the special character.

Note: ExcludeQueues must NOT exceed 2048 characters.

To authorize MQME decryption by application name, you need 2 keywords in the IniFile:

- **UseExcludeQueues** activates this feature. If it has a value of Y, MQME will exclude queues from the encryption/decryption process.
- **ExcludeQueues** specifies the list of queue names that are to be excluded.

```
UseExcludeQueues=Y
ExcludeQueues=SYSTEM*;ABC[0-9][a-f]
```

4.8 ExcludeTopics

This section describes how to explicitly exclude Topics from the encryption/decryption processing when an application has connected to the queue manager. MQME will look up the regular expression patterns from the **ExcludeTopics** keyword in order to determine if the Topic name matches any of the specified expression patterns.

If there is a match, the Topic is excluded from encryption/decryption process. Each regular expression pattern is separated from the next pattern by a semi-colon (';').

In the regular expression pattern:

- '*' matches any sequence of characters (zero or more)
- '?' matches any single character
- '#' matches any single numeric digit (0-9)
- '@' matches any single alphabetic character (A-Z, a-z)
- [SET] matches any of the characters in the specified set
- [!SET] or [^SET] matches any character except those specified in the set (negation).

A SET can be composed of characters or ranges. A range is in the form: 'character – character' (i.e. 0-9 or A-Z). Although this is the simplest range allowed in the [ ] pattern, more complex inclusive ranges such as [0-9a-zA-Z] are allowed. [0-9a-zA-Z] specifies that the character can be 0 through 9 **or** a through z **or** A through Z. Other characters are allowed (ie. 8 bit characters) if your system supports them.

In order to suppress the special syntactic significance of any of these characters '[] * ? # @ ! ^ - \', a backslash ('\') must precede the special character.

Note: ExcludeTopics must NOT exceed 2048 characters.

To authorize MQME decryption by application name, you need 2 keywords in the IniFile:

- **UseExcludeTopics** activates this feature. If it has a value of Y, MQME will exclude Topics from the encryption/decryption process.
- **ExcludeTopics** specifies the list of Topic names that are to be excluded.

```
UseExcludeTopics=Y
ExcludeTopics=test/ABC*;test/XYZ[0-9][a-f]
```

4.9 ExcludeUserIDs

This section describes how to explicitly exclude UserIds from the encryption/decryption processing when a UserId has connected to the queue manager. MQME will look up the regular expression patterns from the **ExcludeUserIDs** keyword in order to determine if the UserId matches any of the specified expression patterns.

If there is a match, the UserId is excluded from encryption/decryption process. Each regular expression pattern is separated from the next pattern by a semi-colon (';').

In the regular expression pattern:

- '*' matches any sequence of characters (zero or more)
- '?' matches any single character
- '#' matches any single numeric digit (0-9)
- '@' matches any single alphabetic character (A-Z, a-z)
- [SET] matches any of the characters in the specified set
- [!SET] or [^SET] matches any character except those specified in the set (negation).

A SET can be composed of characters or ranges. A range is in the form: 'character – character' (i.e. 0-9 or A-Z). Although this is the simplest range allowed in the [ ] pattern, more complex inclusive ranges such as [0-9a-zA-Z] are allowed. [0-9a-zA-Z] specifies that the character can be 0 through 9 **or** a through z **or** A through Z. Other characters are allowed (ie. 8 bit characters) if your system supports them.

In order to suppress the special syntactic significance of any of these characters '[] * ? # @ ! ^ - \', a backslash ('\') must precede the special character.

Note: ExcludeUserIDs must NOT exceed 2048 characters.

To authorize MQME decryption by UserId name, you need 2 keywords in the IniFile:

- **UseExcludeUserIDs** activates this feature. If it has a value of Y, MQME will exclude UserIds from the encryption/decryption process.
- **ExcludeUserIDs** specifies the list of UserId names that are to be excluded.

```
UseExcludeUserIDs=Y
ExcludeUserIDs=fred;barney;wilma;betty
```

4.10 EncryptRFH2Header

EncryptRFH2Header specifies that the MQRFH2 header fields and folders are to be encrypted when the message payload is encrypted. EncryptRFH2Header supports 2 values [Y / N]. The default value is Y.

```
EncryptRFH2Header=Y
```

4.11 PartnerEnabled

PartnerEnabled indicates that MQ Channel Encryption (MQCE) is used on the channels and that the message payload should not be decrypted. PartnerEnabled supports 2 values [Y / N]. The default value is N.

```
PartnerEnabled=N
```

4.12 ExitPath

This section describes the use of the keyword ExitPath to explicitly set the API exit path. By setting ExitPath to a particular path, the default value for ExitPath is overridden. The defaults are as follows:

For Windows:

```
ExitPath=C:\Capitalware\MQME\
```

For IBM MQ 32-bit on Unix and Linux:

```
ExitPath=/var/mqm/exits/
```

For IBM MQ 64-bit on Unix and Linux:

```
ExitPath=/var/mqm/exits64/
```

For IBM MQ on IBM i:

```
ExitPath=/QIBM/UserData/mqm/mqme/
```

```
ExitPath=D:\Vendor\Capitalware\MQME\
```

4.13 LicenseFile

This section will describe how to have a file that contains all of the user's MQME license keys.

The format of the LicenseFile is similar to an IniFile or properties file where each keyword has an associated value. Each keyword and its value are on a separate line. The format is as follows:

```
QMgrName = License_Key
```

Example:

```
MQA1 = 10M0-AAAA-BBBBBBBB
MQB1 = 10M0-XXXX-CCCCCCCC
```

If the queue manager name is not found in the LicenseFile then the License keyword will be used to retrieve the license key value.

The following are the default values for LicenseFile:

For Windows:

```
LicenseFile=C:\Capitalware\MQME\mqme_licenses.ini
```

For IBM MQ 32-bit on Unix and Linux:

```
LicenseFile=/var/mqm/exits/mqme_licenses.ini
```

For IBM MQ 64-bit on Unix and Linux:

```
LicenseFile=/var/mqm/exits64/mqme_licenses.ini
```

For IBM MQ on IBM i:

```
LicenseFile=/QIBM/UserData/mqm/mqme/mqme_licenses.ini
```

4.14 License Key

This section will describe how to license MQ Message Encryption to a particular queue manager.

Note: The License keyword is not required if the user has implemented the LicenseFile keyword or the License file actually exists in the default location.

Your license will look something like: 10M0-AAAA-BBBBBBBB (Note: This is a sample license only and will NOT work).

```
License=10M0-AAAA-BBBBBBBB
```

4.15 Logging

This section describes the necessary entries to enable MQME to record log information. To enable and control logging, there are 4 keywords in the IniFile:

1. **LogMode** specifies what type of logging the user wishes to have. LogMode supports 4 values [Q/N/V/D] where Q is Quiet, N is Normal, V is Verbose and D is Debug. The default value is N.

2. **LogFile** specifies the location of the log file. The default values are as follows:

   For Windows:
   ```
   LogFile=C:\Capitalware\MQME\mqme.log
   ```

   For IBM MQ 32-bit on Linux:
   ```
   LogFile=/var/mqm/mqme/mqme.log
   ```

   For IBM MQ 64-bit on Unix and Linux:
   ```
   LogFile=/var/mqm/mqme/mqme.log
   ```

   For IBM MQ on IBM i:
   ```
   LogFile=/QIBM/UserData/mqm/mqme/mqme.log
   ```

   Token Replacement for LogFile keyword:
   - **%QM%** - Substitutes the name of the queue manager
   - **%UID%** - Substitutes the UserID
   - **%PID%** - Substitutes the Process ID
   - **%TID%** - Substitutes the Thread ID

3. **RotateLogDaily** specifies whether or not the log files will be rotated on a daily basis. A Y value for ' RotateLogDaily' will activate this feature; otherwise, the log files will left as is. The default value is Y.

   In other words, it is possible to keep up to 9 backup log files. The first connection request after midnight (and not at midnight) will cause it to roll/rotate the log files. If there are already 9 backup log files, the ninth backup log file will be deleted and 8 becomes 9, 7 becomes 8, etc...

4. **BackupLogFileCount** specifies the number of backup log files that should be kept by MQME. The default value is 9. This keyword is only used if RotateLogDaily is set to 'Y'.

5 Activating Encrypted Message Data by Queue

This section describes IniFile keywords to activate MQME encryption/decryption against queues and/or topics of a queue manager

5.1 What Value is to be used for Section Name for a Queue?

The section will describe how to determine what value should be used for the Section Name/Queue Name. MQME uses the "Resolved Queue" value when processing what queue to replicate.

5.1.1 Local Queue

For a local queue, simply use the local queue name for the Section Name/Queue Name.

5.1.2 Cluster Queue

For a cluster queue, simply use the cluster queue name for the Section Name/Queue Name.

5.1.3 Alias Queue

For an alias queue, if the client application opened the queue with:

- MQOD version 1 or 2, the user MUST use the '***QUEUE***' value from the alias queue definition for the Section Name/Queue Name.
- MQOD version 3 or higher, the user MUST use the '***TARGET***' value from the alias queue definition for the Section Name/Queue Name.

Example:

```
DISPLAY QALIAS(TEST.Q1.AL)
     1 : DISPLAY QALIAS(TEST.Q1.AL)
AMQ8409: Display Queue details.
   QUEUE(TEST.Q1.AL)                       TYPE(QALIAS)
   ALTDATE(2012-12-04)                     ALTTIME(15.51.41)
   TARGET(TEST.Q1)                         CLUSNL( )
   CLUSTER( )                              CLWLPRTY(0)
   CLWLRANK(0)                             CUSTOM( )
   DEFBIND(OPEN)                           DEFPRTY(0)
   DEFPSIST(NO)                            DEFPRESP(SYNC)
   DEFREADA(NO)                            DESCR( )
   GET(ENABLED)                            PUT(ENABLED)
   PROPCTL(COMPAT)                         SCOPE(QMGR)
   TARGTYPE(QUEUE)
```

5.1.4 Remote Queue

For a remote queue, if the client application opened the queue with:

- MQOD version 1 or 2, the user MUST use the '***QUEUE***' value from the alias queue definition for the Section Name/Queue Name.
- MQOD version 3 or higher, the user MUST use the '***RNAME***' value from the alias queue definition for the Section Name/Queue Name.

Example:

```
DISPLAY QREMOTE(TEST.Q2.RQ)
     1 : DISPLAY QREMOTE(TEST.Q2.RQ)
AMQ8409: Display Queue details.
   QUEUE(TEST.Q2.RQ)                       TYPE(QREMOTE)
   ALTDATE(2012-12-04)                     ALTTIME(15.51.41)
   CLUSNL( )                               CLUSTER( )
   CLWLPRTY(0)                             CLWLRANK(0)
   CUSTOM( )                               DEFBIND(OPEN)
   DEFPRTY(0)                              DEFPSIST(NO)
   DEFPRESP(SYNC)                          DESCR( )
   PUT(ENABLED)                            RQMNAME(MQWT2)
   RNAME(TEST.Q2)                          SCOPE(QMGR)
   XMITQ(MQWT2.XMIT)
```

5.2 Section Name for a Queue

The section stanza must begin in column 1 with a left square bracket '[' followed by a “Q:” and end with a right square bracket ']'. Between the colon and the right square bracket, the user can explicitly specify a queue name or a regular expression pattern.

The MQME will look up the queue name against the regular expression pattern in order to determine if the queue name matches any of the specified expression.

In the regular expression pattern:

- '*' matches any sequence of characters (zero or more)
- '?' matches any single character
- '#' matches any single numeric digit (0-9)
- '@' matches any single alphabetic character (A-Z, a-z)
- [SET] matches any of the characters in the specified set
- [!SET] or [^SET] matches any character except those specified in the set (negation).

A SET can be composed of characters or ranges. A range is in the form: 'character – character' (i.e. 0-9 or A-Z). Although this is the simplest range allowed in the [ ] pattern, more complex inclusive ranges such as [0-9a-zA-Z] are allowed. [0-9a-zA-Z] specifies that the character can be 0 through 9 **or** a through z **or** A through Z. Other characters are allowed (ie. 8 bit characters) if your system supports them.

In order to suppress the special syntactic significance of any of these characters '[] * ? # @ ! ^ - \', a backslash ('\') must precede the special character.

```
[Q:TEST.HR*]
```

Warning: Do not select the 'SYSTEM.*' queues to be encrypted. If you ever wish to deactivate MQME or remove MQME then it will be impossible, as the system message data will be encrypted. Hence, if you do attempt this, then the only way to recover is to delete and re-create the queue manager.

5.2.1 Authorize UserIds for Reading

This section describes how to authorize a particular UserId or a group of UserIDs, such that in order to have MQME decrypt the message data when it is retrieved from a queue by those authorized applications or users.

MQME will look up the regular expression patterns from the **UserIDsForGet** keyword in order to determine if the UserId matches any of the specified expression patterns. Each regular expression pattern is separated from the next pattern by a semi-colon (';').

In the regular expression pattern:

- '*' matches any sequence of characters (zero or more)
- '?' matches any single character
- '#' matches any single numeric digit (0-9)
- '@' matches any single alphabetic character (A-Z, a-z)
- [SET] matches any of the characters in the specified set
- [!SET] or [^SET] matches any character except those specified in the set (negation).

A SET can be composed of characters or ranges. A range is in the form: 'character – character' (i.e. 0-9 or A-Z). Although this is the simplest range allowed in the [ ] pattern, more complex inclusive ranges such as [0-9a-zA-Z] are allowed. [0-9a-zA-Z] specifies that the character can be 0 through 9 **or** a through z **or** A through Z. Other characters are allowed (ie. 8 bit characters) if your system supports them.

In order to suppress the special syntactic significance of any of these characters '[] * ? # @ ! ^ - \', a backslash ('\') must precede the special character.

Note: UserIds must NOT exceed 2048 characters.

```
UserIDsForGet=xyz*;abc[0-9][a-f]
UseGroupsForGet=N
```

If the UseGroupsForGet keyword is set to 'Y' then the UserIDsForGet keyword is ignored (see next section for more information).

5.2.2 Authorize UserIds for Reading using Groups

This section describes the necessary entries to enable 'read authority' checking against a local OS group or a group file. This feature uses the following four keywords:

- **UseGroupsForGet** keyword controls the use of Groups. Set to 'Y' to allow authorization by either OS or a group file.
- **GroupsForGet** keyword specifies the authorized groups that can connect to the queue manager. Each group is separated from the next by a semi-colon (';').
- **UseGroupFileForGet** keyword controls the use of GroupFile. Set to Y to activate feature.
- **GroupFileForGet** keyword specifies the location of the group file (i.e. groups.ini)

5.2.3 Authorization against Local OS

When UseGroups is set to 'Y' and UseGroupFile is set to 'N' then MQME will query the local OS for the groups listed in the Groups keyword.

Example:

```
UseGroupsForGet=Y
GroupsForGet=grpX;grpZ
UseGroupFileForGet=N
```

5.2.4 Authorization against a Group File

When UseGroupsForGet is set to 'Y' and UseGroupFileForGet is set to 'Y' then MQME will query the specified group file given in GroupFileForGet keyword. This section describes how to implement groups files. The group files are implemented in a similar manner to the way they are implemented in Unix and Linux (i.e. **/etc/group** file).

Below is an example group file:

```
unique_group_name = UserID1;UserID2;UserID3
```

Example:

```
grp1=fred;wilma;pebbles
grp2=barney;betty;bammbamm
grpA=arnold;rockhead;slate;gazoo
grpB=dino;puss;doozy;hoppy
```

The following are the default values for GroupFileForGet:

For Windows:
```
GroupFileForGet=C:\Capitalware\MQME\groups.ini
```

For IBM MQ 32-bit on Unix and Linux:
```
GroupFileForGet=/var/mqm/exits/groups.ini
```

For IBM MQ 64-bit on Unix and Linux:
```
GroupFileForGet=/var/mqm/exits64/groups.ini
```

For IBM MQ on IBM i:
```
GroupFileForGet=/QIBM/UserData/mqm/mqme/groups.ini
```

MQME will check, in order, each group listed in the Groups keyword for a particular UserID. The UserId must exist in one of the groups or else MQME will not allow the connection.

Example:

```
UseGroupsForGet=Y
GroupsForGet=grp1;grp2;grpB
UseGroupFileForGet=Y
GroupFileForGet=C:\Capitalware\MQME\groups.ini
```

5.2.5 Authorize UserIds for Writing

This section describes how to authorize UserIds (wildcards are valid) that are allowed to write (put) messages to the queue.

MQME will look up the regular expression patterns from the **UserIDsForPut** keyword in order to determine if the UserId matches any of the specified expression patterns. Each regular expression pattern is separated from the next pattern by a semi-colon (';').

In the regular expression pattern:

- '*' matches any sequence of characters (zero or more)
- '?' matches any single character
- '#' matches any single numeric digit (0-9)
- '@' matches any single alphabetic character (A-Z, a-z)
- [SET] matches any of the characters in the specified set
- [!SET] or [^SET] matches any character except those specified in the set (negation).

A SET can be composed of characters or ranges. A range is in the form: 'character – character' (i.e. 0-9 or A-Z). Although this is the simplest range allowed in the [ ] pattern, more complex inclusive ranges such as [0-9a-zA-Z] are allowed. [0-9a-zA-Z] specifies that the character can be 0 through 9 **or** a through z **or** A through Z. Other characters are allowed (ie. 8 bit characters) if your system supports them.

In order to suppress the special syntactic significance of any of these characters '[] * ? # @ ! ^ - \', a backslash ('\') must precede the special character.

Note: UserIds must NOT exceed 2048 characters.

```
UserIDsForPut=xyz*;abc[0-9][a-f]
```

5.2.6 Authorize UserIds for Writing using Groups

This section describes the necessary entries to enable 'write authority' checking against a local OS group or a group file. This feature uses the following four keywords:

- **UseGroupsForPut** keyword controls the use of Groups. Set to 'Y' to allow authorization by either OS or a group file.
- **GroupsForPut** keyword specifies the authorized groups that can connect to the queue manager. Each group is separated from the next by a semi-colon (';').
- **UseGroupFileForPut** keyword controls the use of GroupFile. Set to Y to activate feature.
- **GroupFileForPut** keyword specifies the location of the group file (i.e. groups.ini)

5.2.7 Authorization against Local OS

When UseGroups is set to 'Y' and UseGroupFile is set to 'N' then MQME will query the local OS for the groups listed in the Groups keyword.

Example:

```
UseGroupsForPut=Y
GroupsForPut=grpX;grpZ
UseGroupFileForPut=N
```

5.2.8 Authorization against a Group File

When UseGroupsForPut is set to 'Y' and UseGroupFileForPut is set to 'Y' then MQME will query the specified group file given in GroupFileForPut keyword. This section describes how to implement groups files. The group files are implemented in a similar manner to the way they are implemented in Unix and Linux (i.e. **/etc/group** file).

Below is an example group file:

```
unique_group_name = UserID1;UserID2;UserID3
```

Example:

```
grp1=fred;wilma;pebbles
grp2=barney;betty;bammbamm
grpA=arnold;rockhead;slate;gazoo
grpB=dino;puss;doozy;hoppy
```

The following are the default values for GroupFileForPut:

For Windows:
```
GroupFileForPut=C:\Capitalware\MQME\groups.ini
```

For IBM MQ 32-bit on Unix and Linux:
```
GroupFileForPut=/var/mqm/exits/groups.ini
```

For IBM MQ 64-bit on Unix and Linux:
```
GroupFileForPut=/var/mqm/exits64/groups.ini
```

For IBM MQ on IBM i:
```
GroupFileForPut=/QIBM/UserData/mqm/mqme/groups.ini
```

MQME will check, in order, each group listed in the Groups keyword for a particular UserID. The UserId must exist in one of the groups or else MQME will not allow the connection.

Example:

```
UseGroupsForPut=Y
GroupsForPut=grp1;grp2;grpB
UseGroupFileForPut=Y
GroupFileForPut=C:\Capitalware\MQME\groups.ini
```

5.2.9 Authorize Application Name for Reading

This section describes how to add a secondary level of authorization. MQME will first check the UserId and then the application name. If both are authorized then MQME will decrypt the message data when the message data is retrieved from a queue.

MQME will look up the regular expression patterns from the **ApplicationsForGet** keyword in order to determine if the application name matches any of the specified expression patterns. Each regular expression pattern is separated from the next pattern by a semi-colon (';').

In the regular expression pattern:

- '*' matches any sequence of characters (zero or more)
- '?' matches any single character
- '#' matches any single numeric digit (0-9)
- '@' matches any single alphabetic character (A-Z, a-z)
- [SET] matches any of the characters in the specified set
- [!SET] or [^SET] matches any character except those specified in the set (negation).

A SET can be composed of characters or ranges. A range is in the form: 'character – character' (i.e. 0-9 or A-Z). Although this is the simplest range allowed in the [ ] pattern, more complex inclusive ranges such as [0-9a-zA-Z] are allowed. [0-9a-zA-Z] specifies that the character can be 0 through 9 **or** a through z **or** A through Z. Other characters are allowed (ie. 8 bit characters) if your system supports them.

In order to suppress the special syntactic significance of any of these characters '[] * ? # @ ! ^ - \', a backslash ('\') must precede the special character.

Note: Applications must NOT exceed 2048 characters.

To authorize MQME decryption by application name, you need 2 keywords in the IniFile:

- **UseApplicationsForGet** allows the use of authorization by application name
- **ApplicationsForGet** specifies a list of application names to be checked

```
UseApplicationsForGet=Y
ApplicationsForGet=mq*;hr[0-9][a-f];abc??01
```

5.2.10 Authorize Application Name for Writing

This section describes how to add a secondary level of authorization. MQME will first check the UserId and then the application name. If both are authorized then MQME will encrypt the message data when the message data is written to a queue.

MQME will look up the regular expression patterns from the **ApplicationsForPut** keyword in order to determine if the application name matches any of the specified expression patterns. Each regular expression pattern is separated from the next pattern by a semi-colon (';').

In the regular expression pattern:

- '*' matches any sequence of characters (zero or more)
- '?' matches any single character
- '#' matches any single numeric digit (0-9)
- '@' matches any single alphabetic character (A-Z, a-z)
- [SET] matches any of the characters in the specified set
- [!SET] or [^SET] matches any character except those specified in the set (negation).

A SET can be composed of characters or ranges. A range is in the form: 'character – character' (i.e. 0-9 or A-Z). Although this is the simplest range allowed in the [ ] pattern, more complex inclusive ranges such as [0-9a-zA-Z] are allowed. [0-9a-zA-Z] specifies that the character can be 0 through 9 **or** a through z **or** A through Z. Other characters are allowed (ie. 8 bit characters) if your system supports them.

In order to suppress the special syntactic significance of any of these characters '[] * ? # @ ! ^ - \', a backslash ('\') must precede the special character.

Note: Applications must NOT exceed 2048 characters.

To authorize MQME decryption by application name, you need 2 keywords in the IniFile:

- **UseApplicationsForPut** allows the use of authorization by application name
- **ApplicationsForPut** specifies a list of application names to be checked

```
UseApplicationsForPut=Y
ApplicationsForPut=mq*;hr[0-9][a-f];abc??01
```

5.2.11 EncryptRFH2Header

EncryptRFH2Header specifies that the MQRFH2 header fields and folders are to be encrypted when the message payload is encrypted. EncryptRFH2Header supports 2 values [Y / N]. The default value is Y.

```
EncryptRFH2Header=Y
```

5.2.12 PartnerEnabled

PartnerEnabled indicates that MQ Channel Encryption (MQCE) is used on the channels and that the message payload should not be decrypted. PartnerEnabled supports 2 values [Y / N]. The default value is N.

```
PartnerEnabled=N
```

5.2.13 KeySize

KeySize specifies a queue specific AES key size used for the encryption/decryption of the message data. Valid values are 128, 192 or 256. The default value is 128.

```
KeySize=256
```

5.2.14 Perform

Perform indicates what functionality MQME will perform. **Perform** supports 3 values [S/E/B]. The default value is E.

- **S** means that MQME will only sign the message
- **E** means that MQME will only encrypt the message
- **B** means that MQME will sign and encrypt the message

When signing the message, MQME creates the digital signature using cryptographic hash function of SHA-2.

```
Perform=S
```

5.2.15 EncPassPhrase & PassPhrase

To enable the use of the user's own PassPhrase, you need 3 keywords in the IniFile:

- **UsePP** allows the use of a user specified PassPhrase
- **EncPassPhrase** specifies an encrypted PassPhrase that will be used for the protected queue. See Appendix C for details on creating the encrypted PassPhrase.
- **PassPhrase** specifies the actual PassPhrase that will be used for the message encryption and/or decryption (can be 16, 24 or 32 characters/digits in length).

What not to use for your PassPhrase:

- A famous quotation from literature, holy books, etc.
- Something easily guessed by intuition

What to use for your PassPhrase:

- A random selection of characters and numbers
- Use a mix of upper and lower characters
- Use special characters like slash, dot, comma, ampersand, etc.

Encrypted PassPhrase (See Appendix C for details on creating the encrypted passphrase.):

```
UsePP=Y
EncPassPhrase=jXzFNIKKwZ52wsQ3CUwqWUBpDaoVRDnLMDkNqhVEOcswMA
```

Plain text PassPhrase:

```
UsePP=Y
PassPhrase=AeKWU31_wky6MZrL
```

Note: If EncPassPhrase keyword is specified then the PassPhrase keyword is ignored.

5.3 Example

5.3.1 Queue Example

If the queue name is called “TEST.Q01” then the user needs to prefix “Q:”, so that the stanza looks like: [Q:TEST.Q01]. Here is an IniFile example with 4 different groups of queues:

```
[Q:TEST.Q01]
KeySize=192
UserIDsForGet=mqtest;tester;mqbrowse
UserIDsForPut=*
UseApplicationsForGet=N
UseApplicationsForPut=N

[Q:TEST.HR*]
KeySize=128
UsePP=Y
PassPhrase=1234567890123456
UserIDsFroGet=mqtest;tester;mqbrowse
UserIDsForPut=*
UseApplicationsForGet=Y
ApplicationsForGet=mrtr.exe;file2msg.exe
UseApplicationsForPut=N

[Q:TEST.PAY*]
KeySize=128
UseGroupsForGet=Y
GroupsForGet=grp1;grp2
UseGroupFileForGet=Y
GroupFileForGet=C:\Capitalware\MQME\groups.ini
UseApplicationsForGet=Y
ApplicationsForGet=mrtr.exe;file2msg.exe
UseApplicationsForPut=N

[Q:TEST.ABC*]
UserIDsForGet=fred;barney;wilma;betty
UserIDsForPut=*
```

6 Activating Encrypted Message Data by Topic

This section describes IniFile keywords to activate MQME encryption/decryption against topics of a queue manager

6.1 What Value is to be used for Section Name for a Topic?

The section will describe how to determine what value should be used for the Section Name/Topic Name. MQME uses the "Resolved Queue" value when processing what queue to replicate.

6.1.1 Topic String

For a topic string, simply use the topic string as the section name.

6.1.2 Topic Object

For a topic object, the user MUST use the resolved topic string.

- If the topic object is defined with a TOPICSTR('test/abc') and the topic object is used as is then the user would use 'test/abc' as the section name.
- If the topic object is defined with a TOPICSTR('test/abc') and the topic object is used with addition topic string of 'extra' then the resolved topic string would be 'test/abc/extra'. Hence, the user would use 'test/abc/extra' as the section name.

6.2 Section Name for a Topic

The section stanza must begin in column 1 with a left square bracket '[' followed by a “T:” and end with a right square bracket ']'. Between the colon and the right square bracket, the user can explicitly specify a topic name or a regular expression pattern.

The MQME will look up the topic string against the regular expression pattern in order to determine if the topic string matches any of the specified expression.

In the regular expression pattern:

- '*' matches any sequence of characters (zero or more)
- '?' matches any single character
- '#' matches any single numeric digit (0-9)
- '@' matches any single alphabetic character (A-Z, a-z)
- [SET] matches any of the characters in the specified set
- [!SET] or [^SET] matches any character except those specified in the set (negation).

A SET can be composed of characters or ranges. A range is in the form: 'character – character' (i.e. 0-9 or A-Z). Although this is the simplest range allowed in the [ ] pattern, more complex inclusive ranges such as [0-9a-zA-Z] are allowed. [0-9a-zA-Z] specifies that the character can be 0 through 9 **or** a through z **or** A through Z. Other characters are allowed (ie. 8 bit characters) if your system supports them.

In order to suppress the special syntactic significance of any of these characters '[] * ? # @ ! ^ - \', a backslash ('\') must precede the special character.

```
[T:test/abc*]
```

Warning: Do not select the '$SYS/*' topics to be encrypted.

6.2.1 Authorize UserIds for Reading

This section describes how to authorize a particular UserId or a group of UserIDs, such that in order to have MQME decrypt the message data when it is retrieved from a queue or topic by those authorized applications or users.

MQME will look up the regular expression patterns from the **UserIDsForGet** keyword in order to determine if the UserId matches any of the specified expression patterns. Each regular expression pattern is separated from the next pattern by a semi-colon (';').

In the regular expression pattern:

- '*' matches any sequence of characters (zero or more)
- '?' matches any single character
- '#' matches any single numeric digit (0-9)
- '@' matches any single alphabetic character (A-Z, a-z)
- [SET] matches any of the characters in the specified set
- [!SET] or [^SET] matches any character except those specified in the set (negation).

A SET can be composed of characters or ranges. A range is in the form: 'character – character' (i.e. 0-9 or A-Z). Although this is the simplest range allowed in the [ ] pattern, more complex inclusive ranges such as [0-9a-zA-Z] are allowed. [0-9a-zA-Z] specifies that the character can be 0 through 9 **or** a through z **or** A through Z. Other characters are allowed (ie. 8 bit characters) if your system supports them.

In order to suppress the special syntactic significance of any of these characters '[] * ? # @ ! ^ - \', a backslash ('\') must precede the special character.

Note: UserIds must NOT exceed 2048 characters.

```
UserIDsForGet=xyz*;abc[0-9][a-f]
UseGroupsForGet=N
```

If the UseGroupsForGet keyword is set to 'Y' then the UserIDsForGet keyword is ignored (see next section for more information).

6.2.2 Authorize UserIds for Reading using Groups

This section describes the necessary entries to enable 'read authority' checking against a local OS group or a group file. This feature uses the following four keywords:

- **UseGroupsForGet** keyword controls the use of Groups. Set to 'Y' to allow authorization by either OS or a group file.
- **GroupsForGet** keyword specifies the authorized groups that can connect to the queue manager. Each group is separated from the next by a semi-colon (';').
- **UseGroupFileForGet** keyword controls the use of GroupFile. Set to Y to activate feature.
- **GroupFileForGet** keyword specifies the location of the group file (i.e. groups.ini)

6.2.3 Authorization against Local OS

When UseGroups is set to 'Y' and UseGroupFile is set to 'N' then MQME will query the local OS for the groups listed in the Groups keyword.

Example:

```
UseGroupsForGet=Y
GroupsForGet=grpX;grpZ
UseGroupFileForGet=N
```

6.2.4 Authorization against a Group File

When UseGroupsForGet is set to 'Y' and UseGroupFileForGet is set to 'Y' then MQME will query the specified group file given in GroupFileForGet keyword. This section describes how to implement groups files. The group files are implemented in a similar manner to the way they are implemented in Unix and Linux (i.e. **/etc/group** file).

Below is an example group file:

```
unique_group_name = UserID1;UserID2;UserID3
```

Example:

```
grp1=fred;wilma;pebbles
grp2=barney;betty;bammbamm
grpA=arnold;rockhead;slate;gazoo
grpB=dino;puss;doozy;hoppy
```

The following are the default values for GroupFileForGet:

For Windows:
```
GroupFileForGet=C:\Capitalware\MQME\groups.ini
```

For IBM MQ 32-bit on Unix and Linux:
```
GroupFileForGet=/var/mqm/exits/groups.ini
```

For IBM MQ 64-bit on Unix and Linux:
```
GroupFileForGet=/var/mqm/exits64/groups.ini
```

For IBM MQ on IBM i:
```
GroupFileForGet=/QIBM/UserData/mqm/mqme/groups.ini
```

MQME will check, in order, each group listed in the Groups keyword for a particular UserID. The UserId must exist in one of the groups or else MQME will not allow the connection.

Example:

```
UseGroupsForGet=Y
GroupsForGet=grp1;grp2;grpB
UseGroupFileForGet=Y
GroupFileForGet=C:\Capitalware\MQME\groups.ini
```

6.2.5 Authorize UserIds for Writing

This section describes how to authorize UserIds (wildcards are valid) that are allowed to write (put) messages to the queue or topic.

MQME will look up the regular expression patterns from the **UserIDsForPut** keyword in order to determine if the UserId matches any of the specified expression patterns. Each regular expression pattern is separated from the next pattern by a semi-colon (';').

In the regular expression pattern:

- '*' matches any sequence of characters (zero or more)
- '?' matches any single character
- '#' matches any single numeric digit (0-9)
- '@' matches any single alphabetic character (A-Z, a-z)
- [SET] matches any of the characters in the specified set
- [!SET] or [^SET] matches any character except those specified in the set (negation).

A SET can be composed of characters or ranges. A range is in the form: 'character – character' (i.e. 0-9 or A-Z). Although this is the simplest range allowed in the [ ] pattern, more complex inclusive ranges such as [0-9a-zA-Z] are allowed. [0-9a-zA-Z] specifies that the character can be 0 through 9 **or** a through z **or** A through Z. Other characters are allowed (ie. 8 bit characters) if your system supports them.

In order to suppress the special syntactic significance of any of these characters '[] * ? # @ ! ^ - \', a backslash ('\') must precede the special character.

Note: UserIds must NOT exceed 2048 characters.

```
UserIDsForPut=xyz*;abc[0-9][a-f]
```

6.2.6 Authorize UserIds for Writing using Groups

This section describes the necessary entries to enable 'write authority' checking against a local OS group or a group file. This feature uses the following four keywords:

- **UseGroupsForPut** keyword controls the use of Groups. Set to 'Y' to allow authorization by either OS or a group file.
- **GroupsForPut** keyword specifies the authorized groups that can connect to the queue manager. Each group is separated from the next by a semi-colon (';').
- **UseGroupFileForPut** keyword controls the use of GroupFile. Set to Y to activate feature.
- **GroupFileForPut** keyword specifies the location of the group file (i.e. groups.ini)

6.2.7 Authorization against Local OS

When UseGroups is set to 'Y' and UseGroupFile is set to 'N' then MQME will query the local OS for the groups listed in the Groups keyword.

Example:

```
UseGroupsForPut=Y
GroupsForPut=grpX;grpZ
UseGroupFileForPut=N
```

6.2.8 Authorization against a Group File

When UseGroupsForPut is set to 'Y' and UseGroupFileForPut is set to 'Y' then MQME will query the specified group file given in GroupFileForPut keyword. This section describes how to implement groups files. The group files are implemented in a similar manner to the way they are implemented in Unix and Linux (i.e. **/etc/group** file).

Below is an example group file:

```
unique_group_name = UserID1;UserID2;UserID3
```

Example:

```
grp1=fred;wilma;pebbles
grp2=barney;betty;bammbamm
grpA=arnold;rockhead;slate;gazoo
grpB=dino;puss;doozy;hoppy
```

The following are the default values for GroupFileForPut:

For Windows:
```
GroupFileForPut=C:\Capitalware\MQME\groups.ini
```

For IBM MQ 32-bit on Unix and Linux:
```
GroupFileForPut=/var/mqm/exits/groups.ini
```

For IBM MQ 64-bit on Unix and Linux:
```
GroupFileForPut=/var/mqm/exits64/groups.ini
```

For IBM MQ on IBM i:
```
GroupFileForPut=/QIBM/UserData/mqm/mqme/groups.ini
```

MQME will check, in order, each group listed in the Groups keyword for a particular UserID. The UserId must exist in one of the groups or else MQME will not allow the connection.

Example:

```
UseGroupsForPut=Y
GroupsForPut=grp1;grp2;grpB
UseGroupFileForPut=Y
GroupFileForPut=C:\Capitalware\MQME\groups.ini
```

6.2.9 Authorize Application Name for Reading

This section describes how to add a secondary level of authorization. MQME will first check the UserId and then the application name. If both are authorized then MQME will decrypt the message data when the message data is retrieved from a queue or topic.

MQME will look up the regular expression patterns from the **ApplicationsForGet** keyword in order to determine if the application name matches any of the specified expression patterns. Each regular expression pattern is separated from the next pattern by a semi-colon (';').

In the regular expression pattern:

- '*' matches any sequence of characters (zero or more)
- '?' matches any single character
- '#' matches any single numeric digit (0-9)
- '@' matches any single alphabetic character (A-Z, a-z)
- [SET] matches any of the characters in the specified set
- [!SET] or [^SET] matches any character except those specified in the set (negation).

A SET can be composed of characters or ranges. A range is in the form: 'character – character' (i.e. 0-9 or A-Z). Although this is the simplest range allowed in the [ ] pattern, more complex inclusive ranges such as [0-9a-zA-Z] are allowed. [0-9a-zA-Z] specifies that the character can be 0 through 9 **or** a through z **or** A through Z. Other characters are allowed (ie. 8 bit characters) if your system supports them.

In order to suppress the special syntactic significance of any of these characters '[] * ? # @ ! ^ - \', a backslash ('\') must precede the special character.

Note: Applications must NOT exceed 2048 characters.

To authorize MQME decryption by application name, you need 2 keywords in the IniFile:

- **UseApplicationsForGet** allows the use of authorization by application name
- **ApplicationsForGet** specifies a list of application names to be checked

```
UseApplicationsForGet=Y
ApplicationsForGet=mq*;hr[0-9][a-f];abc??01
```

6.2.10 Authorize Application Name for Writing

This section describes how to add a secondary level of authorization. MQME will first check the UserId and then the application name. If both are authorized then MQME will encrypt the message data when the message data is written to a queue or topic.

MQME will look up the regular expression patterns from the **ApplicationsForPut** keyword in order to determine if the application name matches any of the specified expression patterns. Each regular expression pattern is separated from the next pattern by a semi-colon (';').

In the regular expression pattern:

- '*' matches any sequence of characters (zero or more)
- '?' matches any single character
- '#' matches any single numeric digit (0-9)
- '@' matches any single alphabetic character (A-Z, a-z)
- [SET] matches any of the characters in the specified set
- [!SET] or [^SET] matches any character except those specified in the set (negation).

A SET can be composed of characters or ranges. A range is in the form: 'character – character' (i.e. 0-9 or A-Z). Although this is the simplest range allowed in the [ ] pattern, more complex inclusive ranges such as [0-9a-zA-Z] are allowed. [0-9a-zA-Z] specifies that the character can be 0 through 9 **or** a through z **or** A through Z. Other characters are allowed (ie. 8 bit characters) if your system supports them.

In order to suppress the special syntactic significance of any of these characters '[] * ? # @ ! ^ - \', a backslash ('\') must precede the special character.

Note: Applications must NOT exceed 2048 characters.

To authorize MQME decryption by application name, you need 2 keywords in the IniFile:

- **UseApplicationsForPut** allows the use of authorization by application name
- **ApplicationsForPut** specifies a list of application names to be checked

```
UseApplicationsForPut=Y
ApplicationsForPut=mq*;hr[0-9][a-f];abc??01
```

6.2.11 EncryptRFH2Header

EncryptRFH2Header specifies that the MQRFH2 header fields and folders are to be encrypted when the message payload is encrypted. EncryptRFH2Header supports 2 values [Y / N]. The default value is Y.

```
EncryptRFH2Header=Y
```

6.2.12 PartnerEnabled

PartnerEnabled indicates that MQ Channel Encryption (MQCE) is used on the channels and that the message payload should not be decrypted. PartnerEnabled supports 2 values [Y / N]. The default value is N.

```
PartnerEnabled=N
```

6.2.13 KeySize

KeySize specifies a queue or topic specific AES key size used for the encryption/decryption of the message data. Valid values are 128, 192 or 256. The default value is 128.

```
KeySize=256
```

6.2.14 Perform

Perform indicates what functionality MQME will perform. **Perform** supports 3 values [S/E/B]. The default value is E.

- **S** means that MQME will only sign the message
- **E** means that MQME will only encrypt the message
- **B** means that MQME will sign and encrypt the message

When signing the message, MQME creates the digital signature using cryptographic hash function of SHA-2.

```
Perform=S
```

6.2.15 EncPassPhrase & PassPhrase

To enable the use of the user's own PassPhrase, you need 3 keywords in the IniFile:

- **UsePP** allows the use of a user specified PassPhrase.
- **EncPassPhrase** specifies an encrypted PassPhrase that will be used for the protected topic. See Appendix C for details on creating the encrypted PassPhrase.
- **PassPhrase** specifies the actual PassPhrase that will be used for the message encryption and/or decryption (can be 16, 24 or 32 characters/digits in length).

What not to use for your PassPhrase:

- A famous quotation from literature, holy books, etc.
- Something easily guessed by intuition

What to use for your PassPhrase:

- A random selection of characters and numbers
- Use a mix of upper and lower characters
- Use special characters like slash, dot, comma, ampersand, etc.

Encrypted PassPhrase (See Appendix C for details on creating the encrypted passphrase.):

```
UsePP=Y
EncPassPhrase=jXzFNIKKwZ52wsQ3CUwqWUBpDaoVRDnLMDkNqhVEOcswMA
```

Plain text PassPhrase:

```
UsePP=Y
PassPhrase=AeKWU31_wky6MZrL
```

Note: If EncPassPhrase keyword is specified then the PassPhrase keyword is ignored.

6.3 Example

6.3.1 Topic Example

If the topic string is called “test/abc” then the user needs to prefix “T:”, so that the stanza looks like: [T:test/abc]. Here is an IniFile example with 4 different groups of queues:

```
[T:test/abc]
KeySize=192
UserIDsForGet=tester;mqbrowse
UserIDsForPut=*
UseApplicationsForGet=N
UseApplicationsForPut=N

[T:test/abc/*]
KeySize=128
UsePP=Y
PassPhrase=1234567890123456
UserIDsFroGet=mqtest;tester;mqbrowse
UserIDsForPut=*
UseApplicationsForGet=Y
ApplicationsForGet=mrtr.exe;file2msg.exe
UseApplicationsForPut=N

[T:test/abcx/HR*]
KeySize=128
UseGroupsForGet=Y
GroupsForGet=grp1;grp2
UseGroupFileForGet=Y
GroupFileForGet=C:\Capitalware\MQME\groups.ini
UseApplicationsForGet=Y
ApplicationsForGet=mrtr.exe;file2msg.exe
UseApplicationsForPut=N

[T:test/xyz*]
UserIDsForGet=fred;barney;wilma;betty
UserIDsForPut=*
```

7 Appendix A – Summary of IniFile

The sample IniFile below is the mqme.ini file supplied for Windows.

```
[default]
Active=Y
LogMode=N
LogFile=C:\Capitalware\MQME\mqme.log

[Q:TEST.HR*]
UserIDsForGet=fred;barney

[T:test/abc*]
UserIDsForGet=wilma;betty
```

Note: Keywords are case sensitive.

The IniFile supports the following keywords and their values:

Keyword	Description of Server-side keywords
Active	**Active** specifies if MQME is enabled or disabled. Active supports 2 values [Y / N]. The default value is Y. e.g. Active=N
ApplicationsForGet	**ApplicationsForGet** allows for the addition of a secondary level of authorization. The application keyword lists the authorized applications. e.g. ApplicationsForGet=mrtr.exe;file2msg.exe Note: Only used if UseApplicationsForGet is set to 'Y'.
ApplicationsForPUt	**ApplicationsForPut** allows for the addition of a secondary level of authorization. The application keyword lists the authorized applications. e.g. ApplicationsForPut=mrtr.exe;file2msg.exe Note: Only used if UseApplicationsForPut is set to 'Y'.
BackupLogFileCount	**BackupLogFileCount** specifies the number of backup logfiles that MQME will be keeping. The default value is 9. e.g. BackupLogFileCount=9

Keyword	Description of Server-side keywords
DebugUserID	**DebugUserID** specifies a list of UserId which should have debug logging turned on for. e.g. **DebugUserID**=fred;barney
EncPassPhrase	**EncPassPhrase** specifies the encrypted PassPhrase for the protected queue/topic. See Appendix C for details on creating the encrypted PassPhrase. e.g. EncPassPhrase=jXzFNIKKwZ52wsQ3CUwqWUBpDaoVRDnLMDkNqhVEOcswMA Note: Only used if UsePassPhrase is set to 'Y'.
EncryptRFH2Header	**EncryptRFH2Header** specifies that the MQRFH2 header fields and folders are to be encrypted when the message payload is encrypted. EncryptRFH2Header supports 2 values [Y / N]. The default value is Y. e.g. **EncryptRFH2Header**=Y
ExcludeApplications	**ExcludeApplications** specifies which particular applications should not be included. e.g. **ExcludeApplications**=test*;gadget Note: Only used if **UseExcludeApplications** is set to 'Y'.
ExcludeQueues	**ExcludeQueues** specifies which particular queues should not be included. e.g. ExcludeQueues=TEST.Q01 Note: Only used if UseExcludeQueues is set to 'Y'.
ExcludeTopics	**ExcludeTopics** specifies which particular topics should not be included. e.g. ExcludeTopics=test/abc;test/abc/s* Note: Only used if UseExcludeTopics is set to 'Y'.

Keyword	Description of Server-side keywords
ExcludeUserIDs	**ExcludeUserIDs** specifies which particular UserIds should not be included. e.g. **ExcludeUserIDs**=fred;barney Note: Only used if **UseExcludeUserIDs** is set to 'Y'.
GroupFileForGet	**GroupFileForGet** specifies the location of the file that will be used for UserId authorization for read access. The following are the default values for GroupFileForGet: For Windows: GroupFileForGet=C:\Capitalware\MQME\groups.ini For IBM MQ 32-bit on Unix and Linux: GroupFileForGet=/var/mqm/exits/groups.ini For IBM MQ 64-bit on Unix and Linux: GroupFileForGet=/var/mqm/exits64/groups.ini For IBM MQ on IBM i: GroupFileForGet=/QIBM/UserData/mqm/mqme/groups.ini e.g. GroupFileForGet=/var/mqm/exits64/groups.ini Note: Only used if UseGroupsForGet is set to 'Y'.

Keyword	Description of Server-side keywords
GroupFileForPut	**GroupFileForGet** specifies the location of the file that will be used for UserId authorization for write access. The following are the default values for GroupFileForPut: For Windows: GroupFileForPut=C:\Capitalware\MQME\groups.ini For IBM MQ 32-bit on Unix and Linux: GroupFileForPut=/var/mqm/exits/groups.ini For IBM MQ 64-bit on Unix and Linux: GroupFileForPut=/var/mqm/exits64/groups.ini For IBM MQ on IBM i: GroupFileForPut=/QIBM/UserData/mqm/mqme/groups.ini e.g. GroupFileForPut=/var/mqm/exits64/groups.ini Note: Only used if UseGroupsForPut is set to 'Y'.
GroupsForGet	**GroupsForGet** specifies the groups that are allowed to access the protected queue for reading. e.g. GroupsForGet=grp1;grp2 Note: Only used if UseGroupsForGet is set to 'Y'.
GroupsForPut	**GroupsForPut** specifies the groups that are allowed to access the protected queue for writing. e.g. GroupsForPut=grp1;grp2 Note: Only used if UseGroupsForPut is set to 'Y'.
IniFileRecheckTime	**IniFileRecheckTime** specifies the amount, in seconds, before the IniFile is checked whether it has changed or not. The default value is 60. e.g. **IniFileRecheckTime**=60

Keyword	Description of Server-side keywords
KeySize	**KeySize** specifies the AES key size used for the encryption/decryption of the message data. Valid values are 128, 192 or 256. The default value is 128. e.g. KeySize=128
License	**License** specifies the queue manager's license key. Your license key will look something like: 10S0-AAAA-BBBBBBBB (Note: This is a sample license only and will NOT work). e.g. License=10M0-AAAA-BBBBBBB
LicenseFile	**LicenseFile** specifies the location of License file that contains all of the customer's license keys. The default values for LicenseFile are as follows: For Windows: LicenseFile=C:\Capitalware\MQME\mqme_licenses.ini For IBM MQ 32-bit on Unix and Linux: LicenseFile=/var/mqm/exits/mqme_licenses.ini For IBM MQ 64-bit on Unix and Linux: LicenseFile=/var/mqm/exits64/mqme_licenses.ini For IBM MQ on IBM i: LicenseFile=/QIBM/UserData/mqm/mqme/mqme_licenses.ini e.g. LicenseFile=/var/mqm/exits64/mqme_licenses.ini
LogFile	**LogFile** specifies the location of the log file. The defaults are as follows: For Windows: LogFile=C:\Capitalware\MQME \mqme.log For IBM MQ 32-bit on Linux: LogFile=/var/mqm/mqme/mqme.log For IBM MQ 64-bit on Unix and Linux: LogFile=/var/mqm/mqme/mqme.log For IBM MQ on IBM i: LogFile=/QIBM/UserData/mqm/mqme/mqme.log

Keyword	Description of Server-side keywords
LogMode	**LogMode** specifies what type of logging the user wishes to have. LogMode supports 4 values [Q / N / V / D] where Q is Quiet, N is Normal, V is Verbose and D is Debug. The default value is N. e.g. LogMode=N
PartnerEnabled	**PartnerEnabled** indicates that MQ Channel Encryption (MQCE) is used on the channels and that the message payload should not be decrypted. PartnerEnabled supports 2 values [Y / N]. The default value is N. e.g. **PartnerEnabled**=N
PassPhrase	**PassPhrase** defines the user's unique PassPhrase. The PassPhrase can be one of three sizes: 16, 24 or 32 characters/digits in length. e.g. PassPhrase=QPriiTJmr4j7aQ2P
Perform	**Perform** indicates which functionality that MQME will perform. Perform supports 3 values [S / E / B]. The default value is E. • **S** means that MQME will only sign the message • **E** means that MQME will only encrypt the message • **B** means that MQME will sign and encrypt the message When signing the message, MQME creates the digital signature using cryptographic hash function of SHA-2. e.g. Perform=E
RotateLogDaily	**RotateLogDaily** specifies whether or not daily log file rotation should take place. RotateLogDaily supports 2 values [Y / N]. The default value is Y. e.g. RotateLogDaily=Y
UseApplicationsForGet	**UseApplicationsForGet** allows the user to add a secondary level of authorization by application for read access. UseApplicationsForGet supports 2 values [Y / N]. The default value is N. e.g. UseApplicationsForGet=Y

Keyword	Description of Server-side keywords
UseApplicationsForPut	**UseApplicationsForPut** allows the user to add a secondary level of authorization by application for write access. UseApplicationsForPut supports 2 values [Y / N]. The default value is N. e.g. UseApplicationsForPut=Y
UseExcludeApplications	**UseExcludeApplications** specifies whether or not ExcludeApplications should be activated. **UseExcludeApplications** supports 2 values [Y / N]. The default value is N. e.g. **UseExcludeApplications**=Y
UseExcludeQueues	**UseExcludeQueues** specifies whether or not ExcludeQueues should be activated. UseExcludeQueues supports 2 values [Y / N]. The default value is N. e.g. UseExcludeQueues=Y
UseExcludeTopics	**UseExcludeTopics** specifies whether or not ExcludeTopics should be activated. UseExcludeTopics supports 2 values [Y / N]. The default value is N. e.g. UseExcludeTopics=Y
UseExcludeUserIDs	**UseExcludeUserIDs** specifies whether or not ExcludeQueues should be activated. **UseExcludeUserIDs** supports 2 values [Y / N]. The default value is N. e.g. **UseExcludeUserIDs**=Y
UseGroupsForGet	**UseGroupsForGet** allows the UserId to be authorized against either the local OS group or a group file for read access. UseGroupsForGet supports 2 values [Y / N]. The default value is N. e.g. UseGroupsForGet=Y

Keyword	Description of Server-side keywords
UseGroupsForPut	**UseGroupsForPut** allows the UserId to be authorized against either the local OS group or a group file for write access. UseGroupsForPut supports 2 values [Y / N]. The default value is N. e.g. UseGroupsForPut=Y
UsePP	**UsePP** allows the user to specify their own PassPhrase. UsePP supports 2 values [Y / N]. The default value is N. e.g. UsePP=Y
UserIDFormatting	**UserIDFormatting** specifies how MQME will handle the incoming UserID. UserIDFormatting supports 3 values [A / U / L]. ('As Is, Uppercase and Lowercase). The default value is A. UserIDFormatting=U
UserIDsForGet	**UserIDsForGet** specifies authorized UserIDs, so that MQME will decrypt the message data when the message data is retrieved from a queue by a user. e.g; UserIDsForGet=fred;barney Note: Only used if UseGroupsForGet is set to 'N'.
UserIDsForPut	**UserIDsForPut** specifies authorized UserIds that are allowed to write (put) messages to the protected queue. e.g; UserIDsForPut=fred;barney Note: Only used if UseGroupsForPut is set to 'N'.

8 Appendix B – MQME Upgrade Procedures

To upgrade an existing installation of MQME from an older version to a newer version, please do the following:

8.1.1 Windows Upgrade

- Stop all MQ applications connecting to the queue manager including monitoring tools
- Stop the queue manager using the MQME API Exit
- Backup all MQME IniFiles in the MQME install directory
- If MQME was installed using the Windows Installer then
  - Click the ***Start*** -> ***All Programs*** -> ***Control Panel*** -> ***Add or Remove Programs***, select MQME from the list and click the ***Remove*** button then follow the prompts to remove it
  - Run the **mqme-setup.exe** file from the **Windows** directory to install the new version
- Otherwise, copy the following files (latest version) to the MQME install directory:
  - mqme.dll
  - rotatelog.bat
- Restore the MQME IniFiles if they were altered/deleted.
- Restart the queue manager
- Restart your MQ applications and any monitoring tools

8.1.2 Linux 32-bit Upgrade

- Log in under the mqm account
- Stop all MQ applications connecting to the queue manager including monitoring tools
- Stop the queue manager using the MQME API Exit
- Backup all MQME IniFiles in the MQME install directory
- Copy the appropriate tar file to the ***/var/mqm/*** directory
- Un-tar the contents of the tar file.
  e.g. For Linux, do the following command:
  ```
  tar -xvf  mqme_linux_x86.tar
  ```
- Run the script as follows:
  ```
  ./setmqme.sh
  ```
- Restore the MQME IniFiles if they were altered / deleted.
- Delete the MQME tar file
- Restart the queue manager
- Restart your MQ applications and any monitoring tools

8.1.3 Unix and Linux 64-bit Upgrade

- Stop all MQ applications connecting to the queue manager including monitoring tools
- Stop the queue manager using the MQME API Exit
- Backup all MQME IniFiles in the MQME install directory
- Copy the appropriate tar file to the ***/var/mqm/*** directory
- Un-tar the contents of the tar file.
  e.g. For AIX 7.1 (or higher), do the following command:
  ```
  tar -xvf  mqme_aix71_64.tar
  ```
- Run the script as follows:
  ```
  ./setmqme.sh
  ```
- Restore the MQME IniFiles if they were altered / deleted.
- Delete the MQME tar file
- Restart the queue manager
- Restart your MQ applications and any monitoring tools

8.1.4 IBM i Upgrade

- Stop all MQ applications connecting to the queue manager including monitoring tools
- Stop the queue manager using the MQME API Exit
- Backup all MQME IniFiles in the MQME install directory
- ftp the IBM i files to the IBM i server using the following command:

```
ftp -s:mqme_iseries.ftp  iseries_hostname
```

```
your-IBM i-userid
your-IBM i-password

binary
cd QGPL
put mqme.savf
quote SITE NAMEFMT 1
quit
```

- Log onto the target IBM i server and do the following commands:

```
RSTLIB SAVLIB(MQME) DEV(*SAVF) SAVF(QGPL/MQME)
CLRSAVF FILE(QGPL/MQME)
CHGOBJOWN  OBJ(MQME) OBJTYPE(*LIB)  NEWOWN(QMQM)
```

- Restore the MQME IniFiles if they were altered / deleted.
- Restart the queue manager
- Restart your MQ applications and any monitoring tools

9 Appendix C - Encrypt PassPhrase

The Encrypt PassPhrase ('enc_pp') program is used to encrypt the PassPhrase for EncPassPhrase keyword.

Syntax:

```
enc_pp  plain_text_PassPhrase
```

Where :

- plain_text_PassPhrase is the user's PassPhrase to be encrypted

The plain text PassPhrase can be one of three sizes: 16, 24 or 32 characters/digits in length. No spaces. ***The user MUST make sure that the PassPhrase length matches the KeySize, otherwise, MQME will discard the EncPassPhrase if there is a mismatch.***

KeySize	PassPhrase length
128	16 characters
192	24 characters
256	32 characters

The enc_pp program outputs the encrypted PassPhrase to the user's screen as follows:

```
Encrpyted PassPhrase: jXzFNIKKwZ52wsQ3CUwqWUBpDaoVRDnLMDkNqhVEOcswMA
```

Copy the 46 characters beginning and place them in the IniFile for the EncPassPhrase keyword.

e.g.

```
EncPassPhrase=jXzFNIKKwZ52wsQ3CUwqWUBpDaoVRDnLMDkNqhVEOcswMA
```

9.1 Examples

9.1.1 Windows

To use the ***enc_pp*** program on Windows, open a Command prompt and change the directory to `C:\CapitalWare\MQME\`

```
enc_pp.exe QPriiTJmr4j7aQ2P
```

9.1.2 Linux 32-bit

To use the ***enc_pp*** program on Linux for MQ 32-bit, open a shell prompt and change directory to `/var/mqm/exits/`

```
enc_pp QPriiTJmr4j7aQ2P
```

9.1.3 Unix or Linux 64-bit

To use the ***enc_pp*** program on Unix/Linux for MQ 64-bit, open a shell prompt and change directory to `/var/mqm/exits64/`

```
enc_pp QPriiTJmr4j7aQ2P
```

9.1.4 IBM i

To use the enc_server program on IBM i for MQ, open a shell prompt (`QSH`) and change directory to `/QIBM/UserData/mqm/mqme/`

```
enc_pp QPriiTJmr4j7aQ2P
```

10 Appendix D – Encryption

MQ Message Encryption Solution uses the Advanced Encryption Standard (AES) to encrypt the message data while it resides in a queue (i.e. data at rest).

Wikipedia

> *the Advanced Encryption Standard (AES) is an encryption standard adopted by the U.S. government. The standard comprises three block ciphers, AES-128, AES-192 and AES-256, adopted from a larger collection originally published as Rijndael. Each AES cipher has a 128-bit block size, with key sizes of 128, 192 and 256 bits, respectively. The AES ciphers have been analyzed extensively and are now used worldwide, as was the case with its predecessor,[3] the Data Encryption Standard (DES).*
>
> *AES was announced by National Institute of Standards and Technology (NIST) as U.S. FIPS PUB 197 (FIPS 197) on November 26, 2001 after a 5-year standardization process in which fifteen competing designs were presented and evaluated before Rijndael was selected as the most suitable (see Advanced Encryption Standard process for more details). It became effective as a Federal government standard on May 26, 2002 after approval by the Secretary of Commerce. It is available in many different encryption packages. AES is the first publicly accessible and open cipher approved by the NSA for top secret information*

11 Appendix E – Digital Signature

MQ Message Encryption Solution uses the SHA-2 to create a cryptographic hash function for the message data.

Wikipedia

> *SHA-2 is a set of cryptographic hash functions (SHA-224, SHA-256, SHA-384, SHA-512) designed by the National Security Agency (NSA) and published in 2001 by the NIST as a U.S. Federal Information Processing Standard. SHA stands for Secure Hash Algorithm. SHA-2 includes a significant number of changes from its predecessor, SHA-1. SHA-2 consists of set of four hash functions with different digest sizes, with 224, 256, 384 or 512 bits respectively.*

12 Appendix F – Capitalware Product Display Version

MQME includes a program to display the product version number. The command to display the product version number is:

`cwdspver`

12.1 Examples

12.1.1 Windows

To use the cwdspver program on Windows, open a Command prompt and change the directory to `C:\Capitalware\MQME\` and type the following:

```
cwdspver.exe
```

12.1.2 Unix and Linux 32-bit

To use the cwdspver program on Unix/Linux for MQ 32-bit, open a shell prompt and change directory to `/var/mqm/exits/` and type the following:

```
./cwdspver
```

12.1.3 Unix and Linux 64-bit

To use the cwdspver program on Unix/Linux for MQ 64-bit, open a shell prompt and change directory to `/var/mqm/exits64/` and type the following:

```
./cwdspver
```

12.1.4 IBM i

To use the cwdspver program on IBM i, issue the following command on the Command Prompt:

```
CALL MQME/CWDSPVER
```

13 Appendix G – Support

The support for MQ Message Encryption can be found at the following location:

By email at:
support@capitalware.com

By regular mail at:

Capitalware Inc.
Attn: MQME Support
Unit 11, 1673 Richmond Street, PMB524
London, Ontario N6G2N3
Canada

14 Appendix H – Summary of Changes

- MQ Message Encryption v4.2.0
  - Added keyword EncryptRFH2Header to handle encrypting the MQRFH2 header or not
  - Added code check if an MQRFH2 message on an XMITQ only has payload data encrypted.
  - Fixed an issue with MQPUT1 not correctly handling the ExcludeQueue.
  - Fixed a buffer overrun condition on MQCALLBACK that would cause the MCA (amqrmppa) process to crash.
  - Fixed an issue in the subroutine that removes trailing blanks
  - Fixed an issue with the logging framework
  - Enhanced the code for dumping the pointers passed into exit.
  - Fixed issue when an invalid or expired license key is used
  - Fixed an issue with default exit path

- MQ Message Encryption v4.1.0
  - Because of APAR IT22258, added code to set the ReturnedLength of MQGMO for MQGET and MQCALLBACK.
  - Erase ExitUserArea field on exiting.
  - Added code to append trailing slash for ExitPath if it is missing.
  - Fixed issue with UserIDFormatting
  - Tuned the logging code

- MQ Message Encryption v4.0.0
  - Added Topic section so that MQME will protect topics.
  - Added UseExcludeTopics and ExcludeTopics keywords to explicitly exclude topics from being protected.
  - Added EncPassPhrase keyword to support the use of encrypted PassPhrase.
  - Added 'enc_pp' program that will create an encrypted PassPhrase.
  - Fixed an issue with determining the application name.
  - Removed MQAPILevel keyword as it is no longer needed.
  - Changed when the authorization is perform. Now it is done during the MQOPEN rather than MQGet and/or MQPUT/1.
  - Fixed an issue in the logging framework where a constant was being modified.

- MQ Message Encryption v3.3.0
  - Added the ability to exclude applications - new keywords: UseExcludeApplications & ExcludeApplications
  - Added the ability to exclude UserIds - new keywords: UseExcludeUserIDs & ExcludeUserIDs
  - Enhanced logging - the LogFile keyword now supports the following tokens: %QM%, %UID%, %PID% & %TID%
  - Added code to clear hostname field before use
  - Fixed memory allocation when MQOPEN has a non-zero reason code

- MQ Message Encryption v3.2.1
  - Added check for flags in the CWME header
  - Fixed an issue with the Ini Processor not finding next section
  - Fixed an issue on Windows with freeing environment variable memory (error with FreeEnvironmentStrings Windows API call)
  - Fixed an issue with using "size_t" variable type when it should have been "int"

- MQ Message Encryption v3.2.0
  - Fixed an issue related to MQ v7.*/v8 and JMS messages

- MQ Message Encryption v3.1.0
  - Added keyword IniFileRecheckTime to only check the IniFile modification time after 'x' seconds
  - Added keyword DebugUserID to selectively enable debugging by UserID
  - Improved the IniFile processing speed.
  - Changed logfile rotation so that it is only done under the mqm UserID
  - Fixed an issue with Enterprise License key not being loaded from a License file.
  - Tested with MQ v8.0
  - Tested with Windows 8/8.1

- MQ Message Encryption v3.0.1
  - Added UserIDFormatting flag to force Lowercase/Uppercase/As_Is UserId formatting on all platforms

- MQ Message Encryption v3.0.0
  - Added support for local OS Group querying
  - Revamped IniFile keywords for 'read authority' for UserId & Applications
    - Removed keywords: UserIDs, UseGroups, Groups, GroupFile, UseApplications and Applications
    - Added new keywords for read authority: UserIDsForGet, UseGroupsForGet, GroupsForGet, UseGroupFileForGet, GroupFileForGet, UseApplicationsForGet and ApplicationsForGet
  - Added new IniFile keywords for 'write authority' for UserId & Applications
    - UserIDsForPut, UseGroupsForPut, GroupsForPut, UseGroupFileForPut, GroupFileForPut, UseApplicationsForPut and ApplicationsForPut
  - Added code in the Ini parser to distinguish between 'ABC' and 'ABCDEF' keywords
  - Increased the accepted IniFile parameter length from 1024 to 2048 characters
  - Added support non-default install for MQ v7.1 & higher multi-install feature on Linux, Unix and Windows
  - Fixed a bug on IBM i (OS/400) related to square bracket ('[' & ']') processing
  - Fixed a bug in the in-memory Ini parser
  - Fixed an issue with BackupLogFileCount
  - Tested with MQ v7.5

- MQ Message Encryption v2.0.0
  - Added new keyword UserIDsForPut which restricts who can put messages to a queue
  - Fixed an issue with MQClose and files being left open
  - Fixed a non-threading issue on AIX
  - Fixed a bug in MQCallback with MQGMO v4 structure
  - Fixed a bug when using Perform=B and a XMIT queue
  - Added code to handle when the data length value passed to the exit is larger than the real data length
  - Updated the logger

- MQ Message Encryption v1.0.0
  - Initial release.

15 Appendix I – License Agreement

This is a legal agreement between you (either an individual or an entity) and Capitalware Inc. By opening the sealed software packages (if appropriate) and/or by using the SOFTWARE, you agree to be bound by the terms of this Agreement. If you do not agree to the terms of this Agreement, promptly return the disk package and accompanying items for a full refund.
SOFTWARE LICENSE

1. GRANT OF LICENSE. This License Agreement (License) permits you to use one copy of the software product identified above, which may include user documentation provided in on-line or electronic form (SOFTWARE). The SOFTWARE is licensed as a single product, to an individual queue manager, or group of queue managers for an Enterprise License. This Agreement requires that each queue manager of the SOFTWARE be Licensed, either individually, or as part of a group. Each queue manager's use of this SOFTWARE must be covered either individually, or as part of an Enterprise License. The SOFTWARE is in use on a computer when it is loaded into the temporary memory (i.e. RAM) or installed into the permanent memory (e.g. hard disk) of that computer. This software may be installed on a network provided that appropriate restrictions are in place limiting the use to registered queue managers only. Each licensed queue manager will be provided with a perpetual license key and the licensee may continue to use the SOFTWARE, so long as the licensee is current on the Yearly Maintenance Fee. If the licensee stops paying the Yearly Maintenance Fee, then the SOFTWARE must be removed from all systems at the end of the current maintenance period.

2. COPYRIGHT. The SOFTWARE is owned by Capitalware Inc. and is protected by United States Of America and Canada copyright laws and international treaty provisions. You may not copy the printed materials accompanying the SOFTWARE (if any), nor print copies of any user documentation provided in on-line or electronic form. You must not redistribute the registration codes provided, either on paper, electronically, or as stored in the files mqme.ini, mqme_licenses.ini or any other form.

3. OTHER RESTRICTIONS. The registration notification provided, showing your authorization code and this License is your proof of license to exercise the rights granted herein and must be retained by you. You may not rent or lease the SOFTWARE, but you may transfer your rights under this License on a permanent basis, provided you transfer this License, the SOFTWARE and all accompanying printed materials, retain no copies, and the recipient agrees to the terms of this License. You may not reverse engineer, decompile, or disassemble the SOFTWARE, except to the extent the foregoing restriction is expressly prohibited by applicable law.

LIMITED WARRANTY

LIMITED WARRANTY. Capitalware Inc. warrants that the SOFTWARE will perform substantially in accordance with the accompanying printed material (if any) and on-line documentation for a period of 365 days from the date of receipt.

CUSTOMER REMEDIES. Capitalware Inc. entire liability and your exclusive remedy shall be, at Capitalware Inc. option, either (a) return of the price paid or (b) repair or replacement of the SOFTWARE that does not meet this Limited Warranty and that is returned to Capitalware Inc.

with a copy of your receipt. This Limited Warranty is void if failure of the SOFTWARE has resulted from accident, abuse, or misapplication. Any replacement SOFTWARE will be warranted for the remainder of the original warranty period or thirty (30) days, whichever is longer.

NO OTHER WARRANTIES. To the maximum extent permitted by applicable law, Capitalware Inc. disclaims all other warranties, either express or implied, including but not limited to implied warranties of merchantability and fitness for a particular purpose, with respect to the SOFTWARE and any accompanying written materials.

NO LIABILITY FOR CONSEQUENTIAL DAMAGES. To the maximum extent permitted by applicable law, in no event shall Capitalware Inc. be liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or other pecuniary loss) arising out of the use or inability to use the SOFTWARE, even if Capitalware Inc. has been advised of the possibility of such damages.

16 Appendix J – Notices

Trademarks:

AIX, IBM, MQSeries, OS/2 Warp, OS/400, iSeries, MVS, OS/390, WebSphere, IBM MQ and z/OS are trademarks of International Business Machines Corporation.

HP-UX is a trademark of Hewlett-Packard Company.

Intel is a registered trademark of Intel Corporation.

Java, J2SE, J2EE, Sun and Solaris are trademarks of Sun Microsystems Inc.

Linux is a trademark of Linus Torvalds.

Mac OS X is a trademark of Apple Computer Inc.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation.

UNIX is a registered trademark of the Open Group.

WebLogic is a trademark of BEA Systems Inc.