MQ Batch Toolkit Installation and Operation Manual

Capitalware Inc.
Unit 11, 1673 Richmond Street, PMB524
London, Ontario, Canada
N6G 2N3
sales@capitalware.com
http://www.capitalware.com

Last Updated: January 2023.

Table of Contents

1 Introduction

1.1 Overview

MQ Batch Toolkit (MQBT) application allows users to manipulate, monitor and manage messages in a queue of an IBM MQ (formally WebSphere MQ & MQSeries) queue manager from a command-line or shell scripting environment.

MQ Batch Toolkit is an excellent tool for developers, programmers, quality assurance testers, and production support personnel who want to do backup and recovery of messages, stress testing of applications, replaying of messages, searching (grep) a queue for a text string, etc..

MQ Batch Toolkit can run on the following platforms: Linux x86 64-bit, macOS (Mac OS X), Windows 7/8/8.1/10 and Raspberry Pi (ARM). MQ Batch Toolkit is able to connect to local queue managers (residing on the same box) or to any remote queue manager.

The remote queue managers can be on any platform that supports distributed queuing including: AIX, HP-UX, HPE NonStop, Linux, IBM i (OS/400), Oracle Solaris, Raspberry Pi (ARM), Tandem, Windows 2008/2012/2016 Server, Windows 7/8/8.1/10 and z/OS (OS/390).

MQ Batch Toolkit has full language support for the following 55 languages: Amharic, Arabic, Azerbaijani, Bengali, Cebuano, Chinese (Mandarin China), Chinese (Mandarin Taiwan), Czech, Danish, Dutch, English, Finnish, French, German, Greek, Gujarati, Hausa, Hebrew, Hindi, Hungarian, Igbo, Indonesian, Italian, Japanese, Javanese, Kannada, Korean, Malay, Malayalam, Marathi, Norwegian, Panjabi, Pashto, Persian, Polish, Portuguese, Romanian, Russian, Shona, Sindhi, Spanish, Sundanese, Swahili, Swedish, Tamil, Telugu, Thai, Turkish, Ukrainian, Urdu, Uzbek, Vietnamese, Xhosa, Yoruba and Zulu.

MQBT supports both forms of MQ security:

- SSL/TLS for connecting to remote queue managers.
- 3rd party security exit for connecting to remote queue managers.

MQ Batch Toolkit contains 45 separate functions grouped into 10 categories:

Queue Management

- Backup a Queue to a single file
- Restore a Queue from a single file
- Retrieve a list of queues
- Find a text string (i.e. grep a queue)
- Clear a Queue
- Clear a Queue by Message Id, Correlation Id or both
- Clear a Queue by Time
- Clear a Queue by Matching String

Message Manipulation of a Queue

- Insert messages (optional formats: binary, string, Pub/Sub (RFH) or JMS (RFH2))
- Copy messages
- Duplicate messages
- Forward messages (optional stripping of the Dead Letter Header)
- Delete messages
- Import messages (optional formats: binary, string, Pub/Sub or JMS)
- Export messages to 1 or many files.
- Read messages and display any MQ header (i.e MQMD, MQRFH, MQRFH2, MQDEAD, MQCIH, MQIIH, & MQXMIT)
- Read messages in a Hex display
- Read messages in an EBCDIC Hex display
- Report generates a document (PDF, RTF or HTML) from one or more messages

Topic Management

- BackupTopic to backup messages of a topic to a single file
- RestoreTopic to restore messages from a single file to a topic
- TopicList to generate a list of Topics

Message Manipulation of a Topic

- Publish messages (optional formats: binary, string, Pub/Sub (RFH) or JMS (RFH2))
- Import messages (optional formats: binary, string, Pub/Sub or JMS)
- Export messages to 1 or many files.
- Subscribe to a topic and read messages and display any MQ header (i.e MQMD, MQRFH, MQRFH2, MQDEAD, MQCIH, MQIIH, & MQXMIT)
- Report generates a document (PDF, RTF or HTML) from one or more messages

Email

- Get an email message and put it on an MQ queue as an MQ Message
- Send an email using the contents of an MQ message

MQ Tools

- CheckUp verifies that attributes of MQ objects are valid and exists.
- PortScan scans TCP ports logging open, insecure MQ ports

Monitoring Tools

- Channel Monitor
- Event Monitor
- Queue Monitor
- Queue Statistics Monitor
- Queue Status Monitor
- Topic Monitor
- Subscription Monitor

Stress Testing Tools

- Get Server
- Put Server
- SIM Client
- SIM Server
- Subscribe Server
- Publish Server

Queue Manager Access Profile

- Add a Queue Manager
- Alter a Queue Manager
- Delete a Queue Manager
- List a Queue Manager

Other

- Register

1.2 Prerequisites

This section provides the minimum supported software levels.

1.2.1 Java

MQ Batch Toolkit requires Java SE 7 or higher.

1.2.2 IBM MQ

MQ Batch Toolkit requires IBM MQ (formally WebSphere MQ & MQSeries) v7.1 or higher.

1.2.3 Operating System

MQ Batch Toolkit is capable of running on the following operating system platforms:

- Windows 7/8/8.1/10
- Linux x86 64-bit (Desktop)
- macOS (Mac OS X) 10.8 or higher (client mode only)
- Raspberry Pi (ARM)

2 Installing MQ Batch Toolkit

This section describes how to install Capitalware's MQ Batch Toolkit.

2.1 Installation

2.1.1 Windows Installation

To install MQ Batch Toolkit on Windows, do the following instructions:

- Run the install program called: **mqbt-setup-withjre.exe**
- The installer follows the standard Windows install procedures and provides default values for the user.
- When the install program has completed execution, there will be a newly created folder under ***Start*** -> ***All Programs*** called ***MQ Batch Toolkit***. This will open a Windows Command prompt.

2.1.2 Linux x86 64-bit Installation

To install MQ Batch Toolkit on Linux x86 64-bit, do the following instructions:

- First, the user needs to set the 'execute' permission for the installer program:

```
chmod +x mqbt-setup-linux.bin
```

- Next, run the installer with the following command:

```
./mqbt-setup-linux.bin
```

- The installer will prompt the user for the installation directory. The default value will be 'Capitalware/MQBT/'.

- To run MQ Batch Toolkit, go to the installation directory and issue the following command:

```
./mqbt
```

2.1.3 macOS Installation

To install MQ Batch Toolkit on macOS (Mac OS X), do the following:

1. ftp or copy the selected mqbt-macOS.tar.zip file to the target platform
2. Unzip the mqbt-macOS.tar.zip to an appropriate directory with the following commands:

   ```
   unzip mqbt-macOS.tar.zip
   tar -xvf mqbt-macOS.tar
   ```

3. Change directory to ***Capitalware/MQBT/***
4. To run MQ Batch Toolkit, go to the installation directory and issue the following command:

   ```
   ./mqbt
   ```

2.1.4 Raspberry Pi Installation

To install MQ Batch Toolkit on Raspberry Pi, do the following:

5. ftp or copy the selected mqbt-RaspberryPi.tar.zip file to the target platform
6. Unzip the mqbt-RaspberryPi.tar.zip to an appropriate directory with the following commands:

   ```
   unzip mqbt-RaspberryPi.tar.zip
   tar -xvf mqbt-RaspberryPi.tar
   ```

7. Change directory to ***Capitalware/MQBT/***
8. To run MQ Batch Toolkit, go to the installation directory and issue the following command:

   ```
   ./mqbt
   ```

2.2 Configuring the Property files

MQ Batch Toolkit contains 2 property files: MQBT.ini and log4j.properties

2.2.1 MQBT.ini

The MQBT.ini file contains global parameters for MQ Batch Toolkit. The MQBT.ini file can also support parameters for specific MQ Batch Toolkit functions (i.e. QList). If a function specific section if included in the MQBT.ini then those parameter values will be used rather than the global parameters. Note: A user can override either the global or function specific values by specifying parameters on the command prompt.

Sample MQBT.ini file

```
[Defaults]
CommProfileDirectory=c:\CommProfileDB.properties
License=0000AAAA0000AAAA
Name=John Doe
Email=john.doe@acme.com

[QList]
CommProfileDirectory=d:\test\CommProfileDB.properties
LogFile=log/qlist.log
```

2.2.2 Global Parameters

MQ Batch Toolkit currently supports 3 global parameters: CommProfileDirectory and License.

- ***CommProfileDirectory***

This parameter contains the full path and filename of a Queue Manager Access Profile file. i.e. c:\test\CommProfileDB.properties

- ***License***

This parameter contains the license key for a registered user of MQ Batch Toolkit.

- ***Email***

This parameter contains the end-user's email address, so that Access Code will be emailed to the user for activation.

- ***Name***

This parameter contains the end-user's name, so that Access Code will be emailed to the user for activation.

- ***ProxyServer***
- ***ProxyPort***

These parameter are used when your company requires users to uses a proxy server to access the internet. The proxy server and port number will be the same values as used in Internet Explorer, Firefox or Chrome browsers.

- ***QueuePrefix***

This parameter is used when create temporary dynamic queue on a z/OS queue manager. The default value is 'CSQ'.

2.2.3 log4j.properties

The log4j.properties file contains parameters related to MQ Batch Toolkit's use of Log4J. The default install of MQ Batch Toolkit contains a log4j.properties file that writes a full log record to the ***log\ mqbt.log*** file plus a partial log record is written to the console (screen / terminal).

Sample log4j.properties file

```
#
# MQBT's log4.properties file
#
# Use two appenders to write to both the log file and the console
#
log4j.category.com.capitalware.mqbt.MQBT=DEBUG, mqbt, stdout
#
# Use one appender to write to the log file only
#
##log4j.category.com.capitalware.mqbt.MQBT=DEBUG, mqbt
#
# "mqbt" appender writes to a file
log4j.appender.mqbt=org.apache.log4j.RollingFileAppender
log4j.appender.mqbt.File=log/mqbt.log
log4j.appender.mqbt.MaxFileSize=1000KB
log4j.appender.mqbt.MaxBackupIndex=9
log4j.appender.mqbt.layout=org.apache.log4j.PatternLayout
log4j.appender.mqbt.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p - %m%n
#
# stdout
#
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%m%n
```

All Log4J output (regardless of the stdout setting) is written to the MQ Batch Toolkit log file. The location and the name of the log file is controlled by the log4j.appender.LF.File property field. The default value for log4j.appender.LF.File is log/mqbt.log

On Windows, this means that if the user installed MQ Batch Toolkit in the default directory of C:\ Capitalware\MQBT\

```
C:\Capitalware\MQBT\log\mqbt.log
```

2.3 Registration

This section will describe how to obtain a license and then register your copy of MQ Batch Toolkit. To purchase a copy of MQ Batch Toolkit (obtain a license), go to the following web site (www.capitalware.com) and select MQ Batch Toolkit, then click ***Buy It.***

2.3.1 Registration Setup

After you have purchased a license, you will be emailed your license. Follow the following instructions to input your license to MQ Batch Toolkit.

Edit the MQBT.ini file located in the products home directory.
i.e.
On Windows:

```
C:\Capitalware\MQBT\MQBT.ini
```

On Linux and macOS (Mac OS X):

```
Capitalware/MQBT/MQBT.ini
```

Your license will look something like: 0000AAAA0000AAAA (Note: This is a sample license only and will NOT work).

In the [Defaults] section of the MQBT.ini file, insert your license key as follows:

```
[Defaults]
License=0000AAAA0000AAAA
Name=John Doe
Email=john.doe@acme.com
```

Save the file and then run MQ Batch Toolkit's Register command to register MQ Batch Toolkit and retrieve an Access Code from Capitalware's web server. Once the registration process is done, the user will be able to define and use an unlimited number of queue managers.

If your company uses a Proxy Server then the user will need to input this information, so that MQ Batch Toolkit can access the internet. Add the ProxyHost and ProxyPort keywords to the MQBT.ini file.

```
[Defaults]
License=0000AAAA0000AAAA
Name=John Doe
Email=john.doe@acme.com
ProxyServer=proxy.acme.com
ProxyPort=8080
```

2.3.2 Interactive Registration

Once the user has inputted the license key, name and email address in the MQBT.ini file, they will need to run the following command to register the license key and retrieve the Access Code:

```
mqbt Register
```

The registration process will create '***mqbt.license.properties***' file in the MQBT installation directory which contains the Access Code.

2.3.3 Connectivity Issues

This section will describe how to handle registering MQ Batch Toolkit when the user has connectivity issues. Before continuing, please make sure you have reviewed the Proxy Server settings in the previous section.

If MQ Batch Toolkit cannot make a connection to Capitalware's web server (with or without a Proxy Server), then the user can email the information to 'support@capitalware.com'.

Email your ***mqbt.log*** file located in the '***log***' directory of MQBT installation directory, ***as an attachment***, to 'support@capitalware.com'.   We will email you your '***mqbt.license.properties***' file.

2.3.4 A New or Replacement PC or Laptop

This section will describe how to handle inputting your information into MQ Batch Toolkit for a new/replacement PC/laptop after you have previously registered your MQ Batch Toolkit license key.

It is very common for users to get a new PC/laptop or have Windows re-installed on their PC/laptop. The first thing the user needs to do is find the email called 'MQ Batch Toolkit Access Code' that they received when they originally registered MQ Batch Toolkit. It will contain their Access Code.

- First, the user needs to follow the installation instructions in section 2.2.1.
- Next, the user will want to copy the following 3 files to their new PC/laptop from their 'home' directory (i.e. C:\Users\{UserId}\ ):
    - CommProfileDB.properties
    - MQBT.ini
    - mqbt.license.properties
    - any other *.ini files that the user may have created

3 Queue Manager Access Profile

This chapter will describe how to create, update, test and delete a Queue Manager Access Profile.

3.1 Adding a Queue Manager Access Profile

This section will describe how to add a Queue Manager Access Profile so that you can connect to a local or remote queue manager.

3.1.1 Purpose

Use the **AddProfile** command to add a Queue Manager Access Profile to a CommProfileDB.properties file. If the CommProfileDB.properties file does not exit then it will be created. A Queue Manager Access Profile contains the necessary information to allow MQ Batch Toolkit to connect to a queue manager.

3.1.2 Syntax

```
mqbt AddProfile [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name
-m Queue_Manager_Name [-h IP_/_HostName] [-n Port_Number] [-c Channel_Name]
[-u Supplied_UserId] [-w Password]
[-x Security_Exit_Class_Name] [-f Jar_File_Location] [-g Security_Exit_Data]
[-i Send_Exit_Class_Name] [-b Jar_File_Location] [-e Send_Exit_Data]
[-r Receive_Exit_Class_Name] [-q Jar_File_Location] [-z Receive_Exit_Data]
[-s SSL_CipherSpec_Name] [-d Peer_Name] [-t TrustStore] [-v
TrustStore_Password] [-k KeyStore] [-j KeyStore_Password] [-l LDAP_Server] [-o
LDAP_Port] [-C] [-F] [-M]
```

3.1.3 Parameters

3.1.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-m	QMgr_Name	The name of the Queue Manager (Note: Queue Manager names are case sensitive.)

3.1.3.2 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-h	IP_or_HostName	The hostname or IP of the server where the queue manager is located.
-n	Port_Number	The port number of the queue manager's listener. Default is 1414.
-c	Channel_Name	The server connection channel (SVRCONN) name.
-u	UserID	UserID to be used when connecting to the queue manager.
-w	Password	Password to be used when connecting to the queue manager.
-x	security_exit_name	The Security Exit class name for the connection. i.e. biz.capitalware.mqausx.MQAUSXJ
-f	path_to_security_exit	The full path to the location of the security exit. i.e. C:\Capitalware\MQAUSX\MQAUSXJ.jar
-g	security_exit_data	Optional: Data for security exit
-i	send_exit_name	The Send Exit name for the connection. i.e. biz.capitalware.mqce.MQCEJ
-b	path_to_exit	The full path to the location of the send exit. i.e. C:\Capitalware\MQCE\MQCEJ.jar
-e	send_exit_data	Optional: Data for send exit
-r	receive_exit_name	The Receive Exit name for the connection. i.e. biz.capitalware.mqce.MQCEJ
-q	path_to_exit	The full path to the location of the receive exit. i.e. C:\Capitalware\MQCE\MQCEJ.jar
-z	receive_exit_data	Optional: Data for receive exit
-s	cipher_spec_name	The CipherSpecName for the SSL connection
-d	distinguished_name	The Distinguished Name to be used on the SSL connection
-t	trusted_store	The TrustedStore to be used on the SSL connection
-v	trusted_store_passwd	The TrustedStore Passwd to be used on the SSL connection
-k	keystore	The Keystore to be used on the SSL connection
-j	keystore_passwd	The Keystore Passwd to be used on the SSL connection
-l	ldap_server	The Ldap Server to be used on the SSL connection
-o	ldap_port	The Ldap Server port number to be used on the SSL connection
-C		Compatibility mode: send the UserId and Password as done in releases prior to IBM MQ V8.
-F		Set if SSL/TLS FIPS required.
-M		Set if the queue manager resides on the mainframe (z/OS).

3.1.3.3 IBM MQ MI (Multi-Instance) or IBM MQ HA (High Availability)

If the remote queue manager is configured for IBM MQ MI (Multi-Instance) or IBM MQ HA (High Availability) then the following rules must be followed for successful connectivity:

- The first hostname or IP address ***MUST*** have the port number explicitly set.
- The second hostname or IP address must ***NOT*** have the port number explicitly set.
- The port value ***MUST*** be set to the correct value for the 2nd hostname or IP address.

If your queue manager is configured to fail over between 2 servers called: serv01 and serv02 and the queue manager's listener is listening on port number 5555 then this is how to set the IP/Hostname and Port # fields:

IP/Hostname:	serv01(5555),serv02
Port #:	5555

3.1.3.4 SSL/TLS CipherSpec Names

SSL/TLS CipherSpec Names	
ANY	ECDHE_RSA_NULL_SHA256
ANY_TLS12	ECDHE_RSA_RC4_128_SHA256
ANY_TLS12_OR_HIGHER	TLS_AES_128_CCM_8_SHA256
ANY_TLS13	TLS_AES_128_CCM_SHA256
ANY_TLS13_OR_HIGHER	TLS_AES_128_GCM_SHA256
ECDHE_ECDSA_3DES_EDE_CBC_SHA256	TLS_AES_256_GCM_SHA384
ECDHE_ECDSA_AES_128_CBC_SHA256	TLS_CHACHA20_POLY1305_SHA256
ECDHE_ECDSA_AES_128_GCM_SHA256	TLS_RSA_WITH_3DES_EDE_CBC_SHA
ECDHE_ECDSA_AES_256_CBC_SHA384	TLS_RSA_WITH_AES_128_CBC_SHA
ECDHE_ECDSA_AES_256_GCM_SHA384	TLS_RSA_WITH_AES_128_CBC_SHA256
ECDHE_ECDSA_NULL_SHA256	TLS_RSA_WITH_AES_128_GCM_SHA256
ECDHE_ECDSA_RC4_128_SHA256	TLS_RSA_WITH_AES_256_CBC_SHA
ECDHE_RSA_3DES_EDE_CBC_SHA256	TLS_RSA_WITH_AES_256_CBC_SHA256
ECDHE_RSA_AES_128_CBC_SHA256	TLS_RSA_WITH_AES_256_GCM_SHA384
ECDHE_RSA_AES_128_GCM_SHA256	TLS_RSA_WITH_DES_CBC_SHA
ECDHE_RSA_AES_256_CBC_SHA384	TLS_RSA_WITH_NULL_SHA256
ECDHE_RSA_AES_256_GCM_SHA384	TLS_RSA_WITH_RC4_128_SHA256

3.1.4 Examples

3.1.4.1 Example 1
Add a profile to allow the MQ Batch Toolkit to connect in bindings mode to the queue manager.

Windows, Linux or macOS
```
mqbt AddProfile -p MQA1 -m MQA1
```

3.1.4.2 Example 2
Add a profile to allow the MQ Batch Toolkit to connect in client mode to the queue manager.

Windows, Linux or macOS
```
mqbt AddProfile -p MQA1 -m MQA1 -h 10.10.10.10 -n 1414 -c TEST.CHL
```

3.2 Altering a Queue Manager Access Profile

This section will describe how to alter a Queue Manager Access Profile so that you can connect to a local or remote queue manager.

3.2.1 Purpose

Use the **AlterProfile** command to update a Queue Manager Access Profile in a CommProfileDB.properties file. A Queue Manager Access Profile contains the necessary information to allow MQ Batch Toolkit to connect to a queue manager.

3.2.2 Syntax

```
mqbt AlterProfile [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name
-m Queue_Manager_Name [-h IP_/_HostName] [-n Port_Number] [-c Channel_Name]
[-u Supplied_UserId] [-w Password]
[-x Security_Exit_Class_Name] [-f Jar_File_Location] [-g Security_Exit_Data]
[-i Send_Exit_Class_Name] [-b Jar_File_Location] [-e Send_Exit_Data]
[-r Receive_Exit_Class_Name] [-q Jar_File_Location] [-z Receive_Exit_Data]
[-s SSL_CipherSpec_Name] [-d Peer_Name] [-t TrustStore] [-v
TrustStore_Password] [-k KeyStore] [-j KeyStore_Password] [-l LDAP_Server] [-o
LDAP_Port] [-C] [-F] [-M]
```

3.2.3 Parameters

3.2.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.

3.2.3.2 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-m	QMgr_Name	The name of the Queue Manager (Note: Queue Manager names are case sensitive.)
-h	IP_or_HostName	The hostname or IP of the server where the queue manager is located.
-n	Port_Number	The port number of the queue manager's listener. Default is 1414.
-c	Channel_Name	The server connection channel (SVRCONN) name.
-u	UserID	UserID to be used when connecting to the queue manager.
-w	Password	Password to be used when connecting to the queue manager.
-x	security_exit_name	The Security Exit class name for the connection. i.e. biz.capitalware.mqausx.MQAUSXJ
-f	path_to_security_exit	The full path to the location of the security exit. i.e. C:\Capitalware\MQAUSX\MQAUSXJ.jar
-g	security_exit_data	Optional: Data for security exit
-i	send_exit_name	The Send Exit name for the connection. i.e. biz.capitalware.mqce.MQCEJ
-b	path_to_exit	The full path to the location of the send exit. i.e. C:\Capitalware\MQCE\MQCEJ.jar
-e	send_exit_data	Optional: Data for send exit
-r	receive_exit_name	The Receive Exit name for the connection. i.e. biz.capitalware.mqce.MQCEJ
-q	path_to_exit	The full path to the location of the receive exit. i.e. C:\Capitalware\MQCE\MQCEJ.jar
-z	receive_exit_data	Optional: Data for receive exit
-s	cipher_spec_name	The CipherSpecName for the SSL connection
-d	distinguished_name	The Distinguished Name to be used on the SSL connection
-t	trusted_store	The TrustedStore to be used on the SSL connection
-v	trusted_store_passwd	The TrustedStore Passwd to be used on the SSL connection
-k	keystore	The Keystore to be used on the SSL connection
-j	keystore_passwd	The Keystore Passwd to be used on the SSL connection
-l	ldap_server	The Ldap Server to be used on the SSL connection
-o	ldap_port	The Ldap Server port number to be used on the SSL connection
-C		Compatibility mode: send the UserId and Password as done in releases prior to IBM MQ V8.
-F		Set if SSL/TLS FIPS required.
-M		Set if the queue manager resides on the mainframe (z/OS).

3.2.3.3 IBM MQ MI (Multi-Instance) or IBM MQ HA (High Availability)

If the remote queue manager is configured for IBM MQ MI (Multi-Instance) or IBM MQ HA (High Availability) then the following rules must be followed for successful connectivity:

- The first hostname or IP address ***MUST*** have the port number explicitly set.
- The second hostname or IP address must ***NOT*** have the port number explicitly set.
- The port value ***MUST*** be set to the correct value for the 2nd hostname or IP address.

If your queue manager is configured to fail over between 2 servers called: serv01 and serv02 and the queue manager's listener is listening on port number 5555 then this is how to set the IP/Hostname and Port # fields:

IP/Hostname:	serv01(5555),serv02
Port #:	5555

3.2.3.4 SSL/TLS CipherSpec Names

SSL/TLS CipherSpec Names	
ANY	ECDHE_RSA_NULL_SHA256
ANY_TLS12	ECDHE_RSA_RC4_128_SHA256
ANY_TLS12_OR_HIGHER	TLS_AES_128_CCM_8_SHA256
ANY_TLS13	TLS_AES_128_CCM_SHA256
ANY_TLS13_OR_HIGHER	TLS_AES_128_GCM_SHA256
ECDHE_ECDSA_3DES_EDE_CBC_SHA256	TLS_AES_256_GCM_SHA384
ECDHE_ECDSA_AES_128_CBC_SHA256	TLS_CHACHA20_POLY1305_SHA256
ECDHE_ECDSA_AES_128_GCM_SHA256	TLS_RSA_WITH_3DES_EDE_CBC_SHA
ECDHE_ECDSA_AES_256_CBC_SHA384	TLS_RSA_WITH_AES_128_CBC_SHA
ECDHE_ECDSA_AES_256_GCM_SHA384	TLS_RSA_WITH_AES_128_CBC_SHA256
ECDHE_ECDSA_NULL_SHA256	TLS_RSA_WITH_AES_128_GCM_SHA256
ECDHE_ECDSA_RC4_128_SHA256	TLS_RSA_WITH_AES_256_CBC_SHA
ECDHE_RSA_3DES_EDE_CBC_SHA256	TLS_RSA_WITH_AES_256_CBC_SHA256
ECDHE_RSA_AES_128_CBC_SHA256	TLS_RSA_WITH_AES_256_GCM_SHA384
ECDHE_RSA_AES_128_GCM_SHA256	TLS_RSA_WITH_DES_CBC_SHA
ECDHE_RSA_AES_256_CBC_SHA384	TLS_RSA_WITH_NULL_SHA256
ECDHE_RSA_AES_256_GCM_SHA384	TLS_RSA_WITH_RC4_128_SHA256

3.2.4 Examples

3.2.4.1 Example 1
Alter an exiting profile to allow the MQ Batch Toolkit to connect in bindings mode to the queue manager.

Windows, Linux or macOS
```
mqbt AlterProfile -p MQA1 -q MQA1
```

3.2.4.2 Example 2
Alter an exiting profile to allow the MQ Batch Toolkit to connect in client mode to the queue manager.

Windows, Linux or macOS
```
mqbt AlterProfile -p MQA1 -q MQA1 -h 10.10.10.10 -n 1414 -c TEST.CHL
```

3.3 Deleting a Queue Manager Access Profile

This section will describe how to delete a Queue Manager Access Profile.

3.3.1 Purpose

Use the **DeleteProfile** command to delete a Queue Manager Access Profile from a CommProfileDB.properties file. A Queue Manager Access Profile contains the necessary information to allow MQ Batch Toolkit to connect to a queue manager.

3.3.2 Syntax

```
mqbt DeleteProfile [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name
```

3.3.3 Parameters

3.3.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.

3.3.3.2 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.

3.3.4 Examples

3.3.4.1 Example 1

Delete a profile from the CommProfileDB.properties file.

Windows, Linux or macOS

```
mqbt DeleteProfile -p MQA1
```

3.4 Listing a Queue Manager Access Profile

This section will describe how to list a Queue Manager Access Profile.

3.4.1 Purpose

Use the **ListProfile** command to display a Queue Manager Access Profile from a CommProfileDB.properties file. A Queue Manager Access Profile contains the necessary information to allow MQ Batch Toolkit to connect to a queue manager.

3.4.2 Syntax

```
mqbt ListProfile [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name
```

3.4.3 Parameters

3.4.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.

3.4.3.2 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.

3.4.4 Examples

3.4.4.1 Example 1

List a profile from a CommProfileDB.properties file.

Windows, Linux or macOS

```
mqbt ListProfile -p MQA1
```

4 Managing a Queue

This chapter will describe the process of selecting, opening and managing an IBM MQ queue.

4.1 Backing up a Queue

This section will describe how to backup all or selected messages in a queue including the MQMD and the message data to a Backup file. A Backup file can be either a SQLite Database or a VEQ formatted file.

4.1.1 Purpose

Use the **Backup** command to write messages from a particular queue to a Backup file. The default is to write the messages to a VEQ formatted file.

4.1.2 Syntax

```
mqbt Backup [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name -q
Queue_Name -f Output_File_Name [-s Start_Position] [-c Message_Count] [-D] [-
A] [-S]
```

4.1.3 Parameters

4.1.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-q	Queue_Name	The name of the queue (Note: Queue names are case sensitive.)
-f	Output_File_Name	The full path and filename of the VEQ output file.

4.1.3.2 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-s	Start_Position	Start position when getting messages from the queue. The default is 1.
-c	Message_Count	Number of messages to be read from the queue. The default is all messages.
-D		Remove messages from the queue as they are read (destructive get).
-A		If this parameter is present then the messages will be appended to the Backup file.
-S		If this parameter is present then the messages will be written to a SQLite database.

4.1.4 Examples

4.1.4.1 Example 1
Backup all messages in the 'TEST01.Q' queue.

Windows
```
mqbt Backup -p MQA1 -q TEST01.Q -f mybackup.veq
```

Linux or macOS
```
mqbt Backup -p MQA1 -q TEST01.Q -f mybackup.veq
```

4.1.4.2 Example 2
Backup 10 messages starting at message number 25 from the 'TEST01.Q' queue and delete them from the queue.

Windows
```
mqbt Backup -p MQA1 -q TEST01.Q -f mybackup.veq -s 25 -c 25 -D
```

Linux or macOS
```
mqbt Backup -p MQA1 -q TEST01.Q -f mybackup.veq -s 25 -c 25 -D
```

4.2 Restoring a Queue

This section will describe how to restore messages to a queue from a Backup file. A Backup file can be either a SQLite Database (*.mqsdb) or a VEQ formatted file (*.veq).

4.2.1 Purpose

Use the **Restore** command to put messages to a particular queue from a Backup file.

4.2.2 Syntax

```
mqbt Restore [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name -q
Queue_Name -f Input_File_Name
```

4.2.3 Parameters

4.2.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-q	Queue_Name	The name of the queue (Note: Queue names are case sensitive.)
-f	Input_File_Name	The full path and filename of the Backup file.

4.2.3.2 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.

4.2.4 Examples

4.2.4.1 Example 1

Restore all messages in the VEQ file to the 'TEST01.Q' queue.

Windows

```
mqbt Restore -p MQA1 -q TEST01.Q -f mybackup.veq
```

Linux or macOS

```
mqbt Restore -p MQA1 -q TEST01.Q -f mybackup.veq
```

4.3 Retrieving a list of Queues

This section will describe how to retrieve a list of queues from a local or remote queue manager.

4.3.1 Purpose

Use the **QList** command to query a queue manager for a current list of queues. The list of queues can content the current queue depth, the queue type, number of input opens (IPPROCS), and number of output opens (IPPROCS). The Qlist function can be used in junction with other scripts for light-weight monitoring of queues of a queue manager.

4.3.2 Syntax

```
mqbt QList [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name [-f
Ouput_File_Name] [-t Queue_Type] [-m Mask] [-D] [-T] [-I] [-O] [-S]
```

4.3.3 Parameters

4.3.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.

4.3.3.2 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-f	Output_File_Name	The full path and filename of the output file.
-t	Queue_Type	The Queue Type can be used to reduce the returned List Of Queues. Valid values are L, A, C and R (Local, Alias, Cluster and Remote). Default is to return all queues.
-m	mask	The mask can be used to reduce the returned List Of Queues. i.e. ABC.*
-D		Along with the List Of Queues, show the current depth of each local queue.
-T		Show the Queue Type for each queue with the List Of Queues.
-I		Show the Input Opens (IPPROCS) for each local queue in the List Of Queues.
-O		Show the Output Opens (OPPROCS) for each local queue in the List Of Queues.
-S		Include the SYSTEM queues in the List Of Queues.

Notes:
If the user does not want to view the extra logging information, then review the Configure Section of Chapter 2 for turning off logging to the console.

4.3.4 Examples

4.3.4.1 Example 1

List all queues in a queue manager.

Windows, Linux or macOS

```
mqbt QList -p MQA1
```

```
INPUT.TEST.QUEUE
MRTR.DEAD.Q
OUTPUT.TEST.QUEUE1
OUTPUT.TEST.QUEUE2
REPLY.Q
REQUEST.QTEST01.Q
TEST02.Q
TEST03.Q
TEST04.Q
TEST05.Q
TEST1
TEST2
TEST3
TEST_BAD_TRIG.Q
TEST_XMIT.Q
```

4.3.4.2 Example 2

List all queues, queue types, current depth, IPPROCS and OPPROCS of a queue manager.

Windows, Linux or macOS

```
mqbt QList -p MQA1 -D -T -I -O
```

```
INPUT.TEST.QUEUE                       QLocal       0    0    0
MRTR.DEAD.Q                            QLocal       0    0    0
OUTPUT.TEST.QUEUE1                     QLocal       0    0    0
OUTPUT.TEST.QUEUE2                     QLocal       0    0    0
REPLY.Q                                QLocal       0    0    0
REQUEST.Q                              QLocal       0    0    0
OTEST01.Q                              QLocal      78    0    0
TEST02.Q                               QLocal       0    0    0
TEST03.Q                               QLocal       0    0    0
TEST04.Q                               QLocal       0    0    0
TEST05.Q                               QLocal      87    0    0
TEST1                                  QLocal       1    0    0
TEST2                                  QLocal      16    0    0
TEST3                                  QLocal      10    0    0
TEST_BAD_TRIG.Q                        QLocal       0    0    0
TEST_XMIT.Q                            QLocal       0    0    0
```

4.3.4.3 Example 3

List only local queues that begin with 'TEST', along with queue types, current depth, IPPROCS and OPPROCS of a queue manager.

Windows, Linux or macOS

```
mqbt QList -p MQA1 -m "TEST0*" -D -T -I -O
```

```
TEST01.Q                                        QLocal   78     0     0
TEST02.Q                                        QLocal    0     0     0
TEST03.Q                                        QLocal    0     0     0
TEST04.Q                                        QLocal    0     0     0
TEST05.Q                                        QLocal   87     0     0
```

4.4 Find

This section will describe how to use the Find feature. The Find function allows the user to search the messages of the queue for a particular text string.

4.4.1 Purpose

Use the **Find** command to search (grep) a queue for a text string.

4.4.2 Syntax

```
mqbt Find [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name -q
Queue_Name -t Text_String [-s Start_Position] [-c Message_Count] [-X] [-I]
```

4.4.3 Parameters

4.4.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-q	Queue_Name	The name of the queue (Note: Queue names are case sensitive.)
-t	Text_String	Text string to be searched for.

4.4.3.2 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-s	Start_Position	Start position when getting messages from the queue. The default is 1.
-c	Message_Count	Number of messages to be read from the queue. The default is all messages.
-I		Perform case insensitive search.
-X		Find messages that do NOT match the text string.

4.4.4 Examples

4.4.4.1 Example 1
Search a queue of a queue manager for a text string.

Windows, Linux or macOS
```
mqbt Find -p MQA1 -t "test"
```

4.4.4.2 Example 2
Search a queue of a queue manager starting at message number 50 for messages without a text string.

Windows, Linux or macOS
```
mqbt Find -p MQA1 -s 50 -t "test" -X
```

4.5 Clearing a Queue

This section will describe how to clear the contents of a queue.

4.5.1 Purpose

Use the **ClearQ** command to clear the messages of a particular queue of a queue manager. If the queue has no input opens and no output opens then the ClearQ function will use the PCF clear command command to clear the queue. Otherwise, it will destructively get each message in the queue to clear the queue.

4.5.2 Syntax

```
mqbt ClearQ [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name -q
Queue_Name
```

4.5.3 Parameters

4.5.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-q	Queue_Name	The name of the queue (Note: Queue names are case sensitive.)

4.5.3.2 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.

4.5.4 Examples

4.5.4.1 Example 1

Clear the messages in a queue of a queue manager.

Windows, Linux or macOS

```
mqbt ClearQ -p MQA1 -q TEST01.Q
```

4.6 Clearing a Queue by ID

This section will describe how to clear messages from a queue that match either a Message ID or Correlation ID or both.

4.6.1 Purpose

Use the **ClearQByID** command to remove the messages of a particular queue of a queue manager that contain particular a Message ID or Correlation ID or both.

4.6.2 Syntax

```
mqbt ClearQByID [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name -q
Queue_Name {[-g Message_ID] [-z Correlation_ID]} [-s Start_Position] [-c
Message_Count]
```

4.6.3 Parameters

4.6.3.1 Required Parameters

Key	Parameter	Description
-q	Queue_Name	The name of the queue (Note: Queue names are case sensitive.)

4.6.3.2 Requires at least 1 of the following parameters:

Key	Parameter	Description
-g	Message_ID	The message id of the message(s) that you wish to delete.
-z	Correlation_ID	The correlation id of the message(s) that you wish to delete.

4.6.3.3 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-s	Start_Position	Start position when getting messages from the queue. The default is 1.
-c	Message_Count	Number of messages to be read from the queue. The default is all messages.
-X		Delete messages that do NOT match either message id or correlation id.

4.6.4 Examples

4.6.4.1 Example 1
Clear the messages with a particular Message ID in a queue of a queue manager.

Windows, Linux or macOS
```
mqbt ClearQByID -p MQA1 -q TEST01.Q -g
414D51204D5141312020202020202020209248C34020000904
```

4.6.4.2 Example 2
Clear the messages with a particular Correlation ID in a queue of a queue manager.

Windows, Linux or macOS
```
mqbt ClearQByID -p MQA1 -q TEST01.Q -z
414243444142434400000000000000000000000000000000
```

4.6.4.3 Example 3
Clear the messages that both a particular Message ID and a particular Correlation ID in a queue of a queue manager.

Windows, Linux or macOS
```
mqbt ClearQByID -p MQA1 -q TEST01.Q -g
414D51204D5141312020202020202020209248C34020000904 -z
414243444142434400000000000000000000000000000000
```

4.7 Clearing a Queue by Time

This section will describe how to clear messages from a queue that are older than "m" minutes and/or "h" hours and/or "d" days.

4.7.1 Purpose

Use the **ClearQByTime** command to remove the messages of a particular queue of a queue manager that are older than a particular time.

4.7.2 Syntax

```
mqbt ClearQByTime [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name -q
Queue_Name {-d Days -h Hours -m Minutes} [-s Start_Position] [-c
Message_Count]
```

4.7.3 Parameters

4.7.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-q	Queue_Name	The name of the queue (Note: Queue names are case sensitive.)

4.7.3.2 Requires at least 1 of the following parameters:

Key	Parameter	Description
-d	Days	Messages older than the days parameter will be deleted.
-h	Hours	Messages older than the hours parameter will be deleted.
-m	Minutes	Messages older than the minutes parameter will be deleted.

4.7.3.3 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-s	Start_Position	Start position when getting messages from the queue. The default is 1.
-c	Message_Count	Number of messages to be read from the queue. The default is all messages.

4.7.4 Examples

4.7.4.1 Example 1

Clear the messages in a queue of a queue manager that are older than 4 hours.

Windows, Linux or macOS
```
mqbt ClearQByTime -p MQA1 -q TEST01.Q -h 4
```

4.7.4.2 Example 2

Clear the messages in a queue of a queue manager that are older than 8 days, 4 hours and 35 minutes.

Windows, Linux or macOS
```
mqbt ClearQByTime -p MQA1 -q TEST01.Q -d 8 -h 4 -m 35
```

4.7.4.3 Example 3

Clear the messages in a queue of a queue manager that are older than 72 hours and 3 minutes.

Windows, Linux or macOS
```
mqbt ClearQByTime -p MQA1 -q TEST01.Q -h 72 -m 3
```

4.8 Clearing a Queue by Matching String

This section will describe how to clear messages from a queue that match a particular string.

4.8.1 Purpose

Use the **ClearQByString** command to clear the messages that contain a text string (or not) from a particular queue of a queue manager.

4.8.2 Syntax

```
mqbt ClearQByString [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name -
q Queue_Name -t Text_String [-s Start_Position] [-c Message_Count] [-I] [-C]
[-X]
```

4.8.3 Parameters

4.8.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-q	Queue_Name	The name of the queue (Note: Queue names are case sensitive.)
-t	Text_String	Text string to be searched for.

4.8.3.2 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-s	Start_Position	Start position when getting messages from the queue. The default is 1.
-c	Message_Count	Number of messages to be read from the queue. The default is all messages.
-I		Perform case insensitive search.
-C		Convert on Get
-X		Delete messages that do NOT match the text string.

4.8.4 Examples

4.8.4.1 Example 1

Clear the messages with a particular text string in messages of a queue of a queue manager.

Windows, Linux or macOS

```
mqbt ClearQByString -p MQA1 -q TEST01.Q -t test
```

4.8.4.2 Example 2

Clear the messages that do not contain a particular text string starting at message number 25 of a queue of a queue manager.

Windows, Linux or macOS

```
mqbt ClearQByString -p MQA1 -q TEST01.Q -t test -s 25 -X
```

5 Managing a Topic

This chapter will describe the process of selecting, opening and managing an IBM MQ topic.

5.1 Backing up a Topic

This section will describe how to backup all or selected messages in a topic including the MQMD and the message data to a Backup file. A Backup file can be either a SQLite Database or a VEQ formatted file.

5.1.1 Purpose

Use the **BackupTopic** command to write messages from a particular topic to a Backup file. The default is to write the messages to a VEQ formatted file.

5.1.2 Syntax

```
mqbt BackupTopic [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name -T
Topic_Name -f Output_File_Name [-s Start_Position] [-c Message_Count] [-D] [-
A] [-S]
```

5.1.3 Parameters

5.1.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-T	Topic_Name	The name of the topic (Note: Topic names are case sensitive.)
-f	Output_File_Name	The full path and filename of the VEQ output file.

5.1.3.2 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-s	Start_Position	Start position when getting messages from the topic. The default is 1.
-c	Message_Count	Number of messages to be read from the topic. The default is all messages.
-D		Remove messages from the topic as they are read (destructive get).
-A		If this parameter is present then the messages will be appended to a Backup file.
-S		If this parameter is present then the messages will be written to a SQLite database.

5.1.4 Examples

5.1.4.1 Example 1
BackupTopic all messages in the 'test/one' topic.

Windows
```
mqbt BackupTopic -p MQA1 -T test/one -f mybackup.veq
```

Linux or macOS
```
mqbt BackupTopic -p MQA1 -T test/one -f mybackup.veq
```

5.1.4.2 Example 2
BackupTopic 10 messages starting at message number 25 from the 'test/one' topic.

Windows
```
mqbt BackupTopic -p MQA1 -T test/one -f mybackup.veq -s 25 -c 10
```

Linux or macOS
```
mqbt BackupTopic -p MQA1 -T test/one -f mybackup.veq -s 25 -c 105
```

5.2 Restoring a Topic

This section will describe how to restore messages to a topic from a Backup file. A Backup file can be either a SQLite Database (*.mqsdb) or a VEQ formatted file (*.veq).

5.2.1 Purpose

Use the **RestoreTopic** command to put messages to a particular topic from a Backup file.

5.2.2 Syntax

```
mqbt RestoreTopic [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name -T
Topic_Name -f Input_File_Name
```

5.2.3 Parameters

5.2.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-T	Topic_Name	The name of the topic (Note: Topic names are case sensitive.)
-f	Input_File_Name	The full path and filename of the Backup file.

5.2.3.2 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.

5.2.4 Examples

5.2.4.1 Example 1

RestoreTopic all messages in the VEQ file to the 'test/one' topic.

Windows
```
mqbt RestoreTopic -p MQA1 -T test/one -f mybackup.veq
```

Linux or macOS
```
mqbt RestoreTopic -p MQA1 -T test/one -f mybackup.veq
```

5.3 Retrieving a list of Topics

This section will describe how to retrieve a list of topics from a local or remote topic manager.

5.3.1 Purpose

Use the **TopicList** command to query a topic manager for a current list of topics. The list of topics can content the current topic depth, the topic type, number of input opens (IPPROCS), and number of output opens (IPPROCS). The TopicList function can be used in junction with other scripts for light-weight monitoring of topics of a topic manager.

5.3.2 Syntax

```
mqbt TopicList [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name [-f
Ouput_File_Name] [-m Mask] [-S]
```

5.3.3 Parameters

5.3.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.

5.3.3.2 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-f	Output_File_Name	The full path and filename of the output file.
-m	mask	The mask can be used to reduce the returned List Of Topics. i.e. test/#
-S		Include the SYSTEM queues in the List Of Topics.

Notes:
If the user does not want to view the extra logging information, then review the Configure Section of Chapter 2 for turning off logging to the console.

5.3.4 Examples

5.3.4.1 Example 1

List all topics in a topic manager.

Windows, Linux or macOS

```
mqbt TopicList -p MQA1
```

```
/Test                                             Yes    No     Allowed   Allowed   0      0
test/ABC/one                                      Yes    No     Allowed   Allowed   0      3
.NULL.                   NULLTOPIC                Yes    No     Allowed   Inhibit   0      0
test                                              Yes    No     Allowed   Allowed   0      0
/Test/Roger              myTopicObject            Yes    No     Allowed   Allowed   0      0
test/ABC                                          Yes    No     Allowed   Allowed   0      0
```

6 Manipulating Messages of a Queue

This chapter will describe the create, update, delete messages in an IBM MQ queue.

6.1 Inserting a Message

This section will describe how to insert a message into a queue of a queue manager.

6.1.1 Purpose

Use the **Insert** command to insert (write) a messages to a particular queue of a queue manager. The message text can be inputted from the command prompt (shell) or be read from a file. The message can be setup in 1 of 4 ways: binary, string, RFH (Pub/Sub) or RFH2 (JMS).

6.1.2 Syntax

```
mqbt Insert [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name -q
Queue_Name {-f Input_File_Name -t Text_String} [-i mqmd_File_Name] [-S] [-P]
[-R] [-r User_Folder] [-1]
```

6.1.3 Parameters

6.1.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-q	Queue_Name	The name of the queue (Note: Queue names are case sensitive.)

6.1.3.2 Requires at least 1 of the following parameters:

Key	Parameter	Description
-f	Input_File_Name	The full path and filename of the input file.
-t	Text_String	Text string to be used as the message data (enclose text string in quotes).

6.1.3.3 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-i	mqmd_File_Name	The full path and filename of the file with the MQMD values
-S		Set the message's MQMD Format field to string (MQSTR).
-P		Set the message's MQMD Format field to RFH (MQRFH version 1).
-R		Set the message's MQMD Format field to RFH2 (MQRFH version 2).
-r	User_Folder	For RFH1 type message, create and set the NameValue pair. For RFH2 type message, create and set the User Folder to the text.
-1		Put each record (line) in the file as a separate message.

6.1.3.4 MQMD IniFile
The description of the MQMD IniFile can be found in Appendix A.

6.1.4 Examples

6.1.4.1 Example 1

Insert a binary message to a queue from a file.

Windows
```
mqbt Insert -p MQA1 -q TEST01.Q -f c:\abc.pdf
```

Linux or macOS
```
mqbt Insert -p MQA1 -q TEST01.Q -f /abc.pdf
```

6.1.4.2 Example 2

Insert a text string from the command prompt as a 'String' message to a queue.

Windows
```
mqbt Insert -p MQA1 -q TEST01.Q -S -t "This is a test message."
```

Linux or macOS
```
mqbt Insert -p MQA1 -q TEST01.Q -S -t "This is a test message."
```

6.1.4.3 Example 3

Insert a message from a file in the Publish/Subscribe format (RFH) to a queue.

Windows
```
mqbt Insert -p MQA1 -q TEST01.Q -P -r "MQPSCommand RegSub MQPSTopic" -f c:\
myfile.txt
```

Linux or macOS
```
mqbt Insert -p MQA1 -q TEST01.Q -P -r "MQPSCommand RegSub MQPSTopic" -f
/myfile.txt
```

6.1.4.4 Example 4

Insert a message from a file in the JMS format (RFH2) to a queue.

Windows
```
mqbt Insert -p MQA1 -q TEST01.Q -R -f c:\myfile.txt
```

Linux or macOS
```
mqbt Insert -p MQA1 -q TEST01.Q -R -f /myfile.txt
```

6.1.4.5 Example 5

Insert a message from a file in the JMS format (RFH2) with a user folder to a queue.

Windows
```
mqbt Insert -p MQA1 -q TEST01.Q -R -f c:\myfile.txt -r
"<usr><up1>ABC</up1></usr>"
```

Linux or macOS
```
mqbt Insert -p MQA1 -q TEST01.Q -R -f /myfile.txt -r
"<usr><up1>ABC</up1></usr>"
```

6.2 Copying a Message

This section will describe how to copy a message from queue to another queue.

6.2.1 Purpose

Use the **Copy** command to 'exactly' copy messages from a source queue to a target queue. The user can you select to have the Dead Letter Header removed if present before the message is written to the target queue.

6.2.2 Syntax

```
mqbt Copy [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name -q
Source_Queue_Name -t Target_Queue_Name [-s Start_Position] [-c Message_Count]
[-X]
```

6.2.3 Parameters

6.2.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-q	Source_Q_Name	The name of the SOURCE queue.
-t	Target_Q_Name	The name of the TARGET queue.

6.2.3.2 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-s	Start_Position	Start position when getting messages from the queue. The default is 1.
-c	Message_Count	Number of messages to be read from the queue. The default is all messages.
-X		Remove any MQ header during the export process

6.2.4 Examples

6.2.4.1 Example 1

Copy all messages from the source queue to the target queue.

Windows, Linux or macOS

```
mqbt Copy -p MQA1 -q TEST01.Q -t TEST02.Q
```

6.2.4.2 Example 2

Copy 50 messages starting at message number 20 from the source queue to the target queue.

Windows, Linux or macOS

```
mqbt Copy -p MQA1 -q TEST01.Q -t TEST02.Q -s 20 -c 50
```

6.3 Duplicating a Message

This section will describe how to duplicate a message in a queue.

6.3.1 Purpose

Use the **Duplicate** command to replicate messages in the queue. The user can you select to have the Dead Letter Header removed if present before the message is written to the target queue.

6.3.2 Syntax

```
mqbt Duplicate [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name -q
Queue_Name [-s Start_Position] [-c Message_Count]
```

6.3.3 Parameters

6.3.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-q	Queue_Name	The name of the queue (Note: Queue names are case sensitive.)

6.3.3.2 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-s	Start_Position	Start position when getting messages from the queue. The default is 1.
-c	Message_Count	Number of messages to be read from the queue. The default is all messages.

6.3.4 Examples

6.3.4.1 Example 1
Duplicate all messages from in the queue.

Windows, Linux or macOS
```
mqbt Duplicate -p MQA1 -q TEST01.Q
```

6.3.4.2 Example 2
Duplicate 50 messages starting at message number 20 in the queue.

Windows, Linux or macOS
```
mqbt Duplicate -p MQA1 -q TEST01.Q -s 20 -c 50
```

6.4 Forward a Message

This section will describe how to forward (move) a message from queue to another queue.

6.4.1 Purpose

Use the **Forward** command to move messages from a source queue to a target queue. The user can you select to have the Dead Letter Header removed if present before the message is written to the target queue.

6.4.2 Syntax

```
mqbt Forward [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name -q
Source_Queue_Name -t Target_Queue_Name [-s Start_Position] [-c Message_Count]
[-X]
```

6.4.3 Parameters

6.4.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-q	Source_Q_Name	The name of the SOURCE queue.
-t	Target_Q_Name	The name of the TARGET queue.

6.4.3.2 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-s	Start_Position	Start position when getting messages from the queue. The default is 1.
-c	Message_Count	Number of messages to be read from the queue. The default is all messages.
-X		Remove any MQ header during the export process

6.4.4 Examples

6.4.4.1 Example 1
Forward all messages from the source queue to the target queue.

Windows, Linux or macOS
```
mqbt Forward -p MQA1 -q TEST01.Q -t TEST02.Q
```

6.4.4.2 Example 2
Forward 50 messages starting at message number 20 from the source queue to the target queue.

Windows, Linux or macOS
```
mqbt Forward -p MQA1 -q TEST01.Q -t TEST02.Q -s 20 -c 50
```

6.5 Deleting a Message

This section will describe how to delete a message from a queue.

6.5.1 Purpose

Use the **Delete** command to delete messages in the queue.

6.5.2 Syntax

```
mqbt Delete [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name -q
Queue_Name [-s Start_Position] [-c Message_Count]
```

6.5.3 Parameters

6.5.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-q	Queue_Name	The name of the queue (Note: Queue names are case sensitive.)

6.5.3.2 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-s	Start_Position	Start position when getting messages from the queue. The default is 1.
-c	Message_Count	Number of messages to be read from the queue. The default is all messages.

6.5.4 Examples

6.5.4.1 Example 1
Delete all messages from in the queue.

Windows, Linux or macOS
```
mqbt Delete -p MQA1 -q TEST01.Q
```

6.5.4.2 Example 2
Delete 50 messages starting at message number 20 in the queue.

Windows, Linux or macOS
```
mqbt Delete -p MQA1 -q TEST01.Q -s 20 -c 50
```

6.6 Importing a File

This section will describe how to import a file and create a message that will be written to a queue.

6.6.1 Purpose

Use the **Import** command to write messages to a particular queue from a text file. The message text will be read from a file. The message can be setup in 1 of 4 ways: binary, string, RFH (Pub/Sub) or RFH2 (JMS).

6.6.2 Syntax

```
mqbt Import [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name -q
Queue_Name -f Input_File_Name [-i mqmd_File_Name] [-S] [-P] [-R] [-r
User_Folder]
```

6.6.3 Parameters

6.6.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-q	Queue_Name	The name of the queue (Note: Queue names are case sensitive.)
-f	Input_File_Name	The full path and filename of the input file. Wildcard '*' can be used.

6.6.3.2 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-i	mqmd_File_Name	The full path and filename of the file with the MQMD values
-S		Set the message's MQMD Format field to string (MQSTR).
-P		Set the message's MQMD Format field to RFH (MQRFH version 1).
-R		Set the message's MQMD Format field to RFH2 (MQRFH version 2).
-r	User_Folder	For RFH1 type message, create and set the NameValue pair. For RFH2 type message, create and set the User Folder to the text.

6.6.3.3 MQMD IniFile

The description of the MQMD IniFile can be found in Appendix A.

6.6.4 Examples

6.6.4.1 Example 1

Import a binary message to a queue from a file.

Windows
```
mqbt Import -p MQA1 -q TEST01.Q -f c:\abc.pdf
```

Linux or macOS
```
mqbt Import -p MQA1 -q TEST01.Q -f /abc.pdf
```

6.6.4.2 Example 2

Import a 'string' message to a queue from a file.

Windows
```
mqbt Import -p MQA1 -q TEST01.Q -S -f textfile.txt
```

Linux or macOS
```
mqbt Import -p MQA1 -q TEST01.Q -S -f textfile.txt
```

6.6.4.3 Example 3

Import a message from a file in the Publish/Subscribe format (RFH) to a queue.

Windows
```
mqbt Import -p MQA1 -q TEST01.Q -P -r "MQPSCommand RegSub MQPSTopic" -f c:\
myfile.txt
```

Linux or macOS
```
mqbt Import -p MQA1 -q TEST01.Q -P -r "MQPSCommand RegSub MQPSTopic" -f
/myfile.txt
```

6.6.4.4 Example 4

Import a message from a file in the JMS format (RFH2) to a queue.

Windows
```
mqbt Import -p MQA1 -q TEST01.Q -R -f c:\myfile.txt
```

Linux or macOS
```
mqbt Import -p MQA1 -q TEST01.Q -R -f /myfile.txt
```

6.6.4.5 Example 5

Import a message from a file in the JMS format (RFH2) with a user folder to a queue.

Windows
```
mqbt Import -p MQA1 -q TEST01.Q -R -f c:\myfile.txt -r
"<usr><up1>ABC</up1></usr>"
```

Linux or macOS
```
mqbt Import -p MQA1 -q TEST01.Q -R -f /myfile.txt -r
"<usr><up1>ABC</up1></usr>"
```

6.7 Export a Message

This section will describe how to export a message to a file from a queue.

6.7.1 Purpose

Use the **Export** command to write messages from a queue to a text file. Reading the messages in the queue can be done either destructively or non-destructively. Also, the 'Convert on Get' option can be specified for the MQGET.

6.7.2 Syntax

```
mqbt Export [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name -q
Queue_Name -f Output_File_Name [-s Start_Position] [-t Trailer_Text] [-c
Message_Count] [-D] [-A] [-X] [-1]
```

6.7.3 Parameters

6.7.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-q	Queue_Name	The name of the queue (Note: Queue names are case sensitive.)
-f	Output_File_Name	The full path and filename of the output file.

6.7.3.2 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-s	Start_Position	Start position when getting messages from the queue. The default is 1.
-c	Message_Count	Number of messages to be read from the queue. The default is all messages.
-t	Trailer_Text	Trailer text string addended after each exported message. Use "\n" to represent CRLF on Windows and LF on Linux or macOS/Linux.
-C		Convert on Get
-D		Remove messages from the queue as they are read (destructive get).
-A		If this parameter is present then the messages will be appended to output file.
-X		Remove any MQ header during the export process
-1		Write all messages to same file.

6.7.4 Examples

6.7.4.1 Example 1

Export all messages in the 'TEST01.Q' queue.

Windows
```
mqbt Export -p MQA1 -q TEST01.Q -f textfile.txt
```

Linux or macOS
```
mqbt Export -p MQA1 -q TEST01.Q -f textfile.txt
```

If there were 8 messages in the queue, then the output files would look like:

```
textfile_0001.txt
textfile_0002.txt
textfile_0003.txt
textfile_0004.txt
textfile_0005.txt
textfile_0006.txt
textfile_0007.txt
textfile_0008.txt
```

6.7.4.2 Example 2

Export 10 messages starting at message number 25 from the 'TEST01.Q' queue and delete them from the queue.

Windows
```
mqbt Export -p MQA1 -q TEST01.Q -f textfile.txt -s 25 -c 25 -D
```

Linux or macOS
```
mqbt Export -p MQA1 -q TEST01.Q -f textfile.txt -s 25 -c 25 -D
```

6.7.4.3 Example 3

Export all messages doing a destructive get (deleting the messages) with the option of 'Convert on Get' from the 'TEST01.Q' queue.

Windows
```
mqbt Export -p MQA1 -q TEST01.Q -f textfile.txt -D -C
```

Linux or macOS
```
mqbt Export -p MQA1 -q TEST01.Q -f textfile.txt -D -C
```

6.7.4.4 Example 4

Export 10 messages from the 'TEST01.Q' queue, appending the messages to a file, appending just a NewLine (CRLF or LF) and delete them from the queue.

Windows
```
mqbt Export -p MQA1 -q TEST01.Q -f textfile.txt -t "\n" -A -D
```

Linux or macOS
```
mqbt Export -p MQA1 -q TEST01.Q -f textfile.txt -t "\n" -A -D
```

6.7.4.5 Example 5

Export 10 messages from the 'TEST01.Q' queue, appending the messages to a file, appending trailer text and delete them from the queue.

Windows
```
mqbt Export -p MQA1 -q TEST01.Q -f textfile.txt -t "\n\n" -A -D
```

Linux or macOS
```
mqbt Export -p MQA1 -q TEST01.Q -f textfile.txt -t "\n\n" -A -D
```

6.8 Reading Messages

This section will describe how to read a message in a queue. The read function has support for well-known WMQ messages formats: MQMD, MQRFH, MQRFH2, MQCIH, MQDEAD, MQIIH, MQXMIT, MQHSAP and SMQBAD.

6.8.1 Purpose

Use the **Read** command to read messages in the queue.

6.8.2 Syntax

```
mqbt Read [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name -q
Queue_Name [-s Start_Position] [-c Message_Count] [-C] [-D] [-M] [-H] [-E]
```

6.8.3 Parameters

6.8.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-q	Queue_Name	The name of the queue (Note: Queue names are case sensitive.)

6.8.3.2 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-s	Start_Position	Start position when getting messages from the queue. The default is 1.
-c	Message_Count	Number of messages to be read from the queue. The default is all messages.
-C		Convert on Get
-D		Remove messages from the queue as they are read (destructive get).
-M		Display the MQMD fields before display the message's data.
-H		Display the message data in a hex format.
-E		Display the message data in a hex format but the character convert to EBCDIC.

6.8.4 Examples: Reading Messages with a Raw Data format

6.8.4.1 Example 1

Read all messages from the queue.

Windows, Linux or macOS

```
mqbt Read -p MQA1 -q TEST01.Q
```

If the queue had 1 message, it may look like this (the message text begins after 'Raw Data:'):

```
--------------------------- Message # 1 ----------------------------
Raw Data:
This is a test message.
Line #2
and the last line - #3
```

6.8.5 Examples: Reading Messages with a Hex Data format

6.8.5.1 Example 1

Read all messages in Hex format from the queue.

Windows, Linux or macOS

```
mqbt Read -p MQA1 -q TEST01.Q -H
```

If the queue had 1 message, it may look like this (the message text begins after 'Hex Data:'):

```
--------------------------- Message # 1 ----------------------------
Hex Data:
  00000000  54 68 69 73 20 69 73 20 61 20 74 65 73 74 20 6D  'This is a test m'
  00000010  65 73 73 61 67 65 2E 0A 4C 69 6E 65 20 23 32 0A  'essage. Line #2 '
  00000020  61 6E 64 20 74 68 65 20 6C 61 73 74 20 6C 69 6E  'and the last lin'
  00000030  65 20 2D 20 23 33 0A                             'e - #3          '
```

6.8.6 Examples: Reading Messages with a EBCDIC Data format

6.8.6.1 Example 1

Read all messages in EBCDIC Hex format from the queue.

Windows, Linux or macOS

```
mqbt Read -p MQA1 -q TEST01.Q -E
```

If the queue had 1 message, it may look like this (the message text begins after 'EBCDIC Data:'):

```
--------------------------- Message # 1 ----------------------------
EBCDIC Data:
  00000000  2E 2E 2E 2E 2E 2E 2E 2E 2F 2E 2E 2E 2E 2E 2E 5F  '......../......_'
  00000010  2E 2E 2E 2F 2E 2E 2E 2E 3C 2E 3E 2E 2E 2E 2E 2E  '.../....<.>.....'
  00000020  2F 3E 2E 2E 2E 2E 2E 2E 25 2F 2E 2E 2E 25 2E 3E  '/>......%/...%.>'
  00000030  2E 2E 2E 2E 2E 2E 2E                             '.......         '
```

6.8.7 Examples: Reading Messages with a MQMD format

6.8.7.1 Example 1
Read all messages from the queue and display the message's MQMD (message descriptor).

Windows, Linux or macOS
```
mqbt Read -p MQA1 -q TEST01.Q -M
```

If the queue had 1 message, it may look like this (the message text begins after 'Raw Data:'):

```
--------------------------- Message # 1 ----------------------------
Message Descriptor:
*General:
  Version: 2                    Message Type: MQMT_DATAGRAM
  Message Priority: 0           Message Persistence: MQPER_NOT_PERSISTENT
  Put Date: 2004-08-30          Expiry Interval: -1
  Put Time: 01:40:12.280        Backout Count: 0
  Reply-To Queue:
  Reply-To Queue Manager:       MQA1
*Report:
  Report: 0                     Original Length: -1
  Feedback Code: MQFB_NONE
*Context:
  User ID:                      'rlacroix    '
  Put-Application Type:         MQAT_WINDOWS_NT
  Put-Application Name:         'C:\WINNT\system32\javaw.exe '
  Application Identity Data:    '                                '
  Application Origin Data:      '    '
  Accounting Token:
X'16010515000000235F636B54EC114F828BA628E80300000000000000000000000B'
*Identifiers:
  Message ID:
X'414D51204D5141312020202020202020F9BB324120000401'
  Correlation ID:
X'000000000000000000000000000000000000000000000000'
  Group ID:
X'000000000000000000000000000000000000000000000000'
*Segmentation:
  Logical Sequence Number: 1
  Offset: 1                     Message Flags: 0
*Attributes:
  Format: MQSTR                 Message Data Length: 55
  CCSID: 819                    Encoding: 273

Raw Data:
This is a test message.
Line #2
and the last line - #3
```

6.8.7.2 Example 2

Read all messages from the queue and display the message's MQMD (message descriptor) with Hex Data.

Windows, Linux or macOS

```
mqbt Read -p MQA1 -q TEST01.Q -M -H
```

If the queue had 1 message, it may look like this (the message text begins after 'Hex Data:'):

```
--------------------------- Message # 1 ----------------------------
Message Descriptor:
*General:
  Version: 2                    Message Type: MQMT_DATAGRAM
  Message Priority: 0           Message Persistence: MQPER_NOT_PERSISTENT
  Put Date: 2004-08-30          Expiry Interval: -1
  Put Time: 01:40:12.280        Backout Count: 0
  Reply-To Queue:
  Reply-To Queue Manager:       MQA1
*Report:
  Report: 0                     Original Length: -1
  Feedback Code: MQFB_NONE
*Context:
  User ID:                      'rlacroix    '
  Put-Application Type:         MQAT_WINDOWS_NT
  Put-Application Name:         'C:\WINNT\system32\javaw.exe '
  Application Identity Data:    '                                '
  Application Origin Data:      '    '
  Accounting Token:
X'16010515000000235F636B54EC114F828BA628E80300000000000000000000000B'
*Identifiers:
  Message ID:                   X'414D51204D5141312020202020202020F9BB324120000401'
  Correlation ID:               X'000000000000000000000000000000000000000000000000'
  Group ID:                     X'000000000000000000000000000000000000000000000000'
*Segmentation:
  Logical Sequence Number: 1
  Offset: 1                     Message Flags: 0
*Attributes:
  Format: MQSTR                 Message Data Length: 55
  CCSID: 819                    Encoding: 273

Hex Data:
  00000000  54 68 69 73 20 69 73 20 61 20 74 65 73 74 20 6D  'This is a test m'
  00000010  65 73 73 61 67 65 2E 0A 4C 69 6E 65 20 23 32 0A  'essage. Line #2 '
  00000020  61 6E 64 20 74 68 65 20 6C 61 73 74 20 6C 69 6E  'and the last lin'
  00000030  65 20 2D 20 23 33 0A                             'e - #3          '
```

6.8.8 Examples: Reading Messages with a MQRFH format

6.8.8.1 Example 1

Read all messages from the queue.

Windows, Linux or macOS

```
mqbt Read -p MQA1 -q TEST01.Q
```

If the queue had 1 message, it may look like this (the message text begins after 'Raw Data:'):

```
--------------------------- Message # 1 ----------------------------
Rules and Formatting Header v1:
  Structure ID: RFH            Version: 1
  Header Length: 134           Format: MQSTR
  CCSID: 819                   Encoding: 273
  Flags: 0
Raw Data:
MQPSCommand  RegSub  MQPSTopic topic MQPSQName subscriberqueue MQPSStreamName
stream MQPSQMgrName MQA1
```

6.8.9 Examples: Reading Messages with a MQRFH2 format

6.8.9.1 Example 1

Read all messages from the queue.

Windows, Linux or macOS
```
mqbt Read -p MQA1 -q TEST01.Q -H
```

If the queue had 1 message, it may look like this (the message text begins after 'Raw Data:'):

```
--------------------------- Message # 1 ----------------------------
Rules and Formatting Header v2:
  Structure ID: RFH            Version: 2
  Header Length: 152           Format: MQSTR
  CCSID: 437                   Encoding: 546
  Name Value CCSID: 1208       Flags: 0
Folder:
<mcd><Msd>jms_text</Msd></mcd>
Folder:
<jms><jmsp111>oneone</jmsp111></jms>
Folder:
<usr><u111>oneoneoneone</u111></usr>
Raw Data:
this is the message body text
```

6.8.10 Examples: Reading Messages with a MQDEAD format

6.8.10.1 Example 1

Read all messages with MQDEAD header from the queue.

Windows, Linux or macOS

```
mqbt Read -p MQA1 -q TEST01.Q
```

If the queue had 1 message, it may look like this (the message text begins after 'Raw Data:'):

```
--------------------------- Message # 1 ----------------------------
Dead Letter Header:
  Reason Code: 265             Reason Text: MQFB_APPL_CANNOT_BE_STARTED
  Destination Q Name:          SYSTEM.DEFAULT.INITIATION.QUEUE
  Destination QMgr Name:       MQA1
  CCSID:                       437
  Encoding:                    546
  Format:                      MQTRIG
  Put-Application Type:        MQAT_WINDOWS_NT
  Put-Application Name:        'RUNMQTRM                    '
  Put Date:                    2004-08-30
  Put Time:                    12:24:56.050

Raw Data:
TM      BAD_TRIG_QUEUE                                  BAD_PROC

            c:\GoToNothing.bat
```

6.8.11 Examples: Reading Messages with a MQCIH format

6.8.11.1 Example 1

Read all messages with MQCIH header from the queue.

Windows, Linux or macOS
```
mqbt Read -p MQA1 -q TEST01.Q
```

If the queue had 1 message, it may look like this (the message text begins after 'Raw Data:'):

```
--------------------------- Message # 1 ----------------------------
CICS Information Header:
  Structure ID: CIH             Version: 1
  Header Length: 164            Format:
  CCSID: 819                    Encoding: 1
  Flags: 0                      Return Code: 0
  Comp Code: 0                  Reason: 0
  UOW Control: 273              Get Wait Interval: -2
  Link Type: 1                  Output Data Length: -1
  Facility Keep Time: 0         ADS Descriptor: 0
  Conversational Task: 0        Task End Status: 0
  Function:                     Facility: X'0000000000000000'
  Abend Code:                   Authenticator:
  Reply To Format:              Remote SysId:
  Remote TransId:               TransactionId:
  Facility Like:                Attention Id:
  Start Code:                   Cancel Code:
  Next TransactionId:           Cursor Position: 0
  Error Offset: 0               Input Item: 0
Raw Data:
```

6.8.12 Examples: Reading Messages with a MQIIH format

6.8.12.1 Example 1

Read all messages with MQIIH header from the queue.

Windows, Linux or macOS

```
mqbt Read -p MQA1 -q TEST01.Q
```

If the queue had 1 message, it may look like this (the message text begins after 'Raw Data:'):

```
--------------------------- Message # 1 ----------------------------
IMS Information Header:
  Structure ID: IIH             Version: 1
  Header Length: 84             Format:
  CCSID: 0                      Encoding: 785
  Flags: 0                      LTerm Override:
  MFS Map Name:                 Reply To Format:
  Authenticator:                Tran State: X'31'
  Commit Mode: X'20'            Security Scope: X'A5'
  Tran InstanceId: X'BA28F0725B3FB16140404040404040'
Raw Data:
+¦¦=@@@@@+¦¦=±+-=··±==)======··±±=±±±=)=˜++++@=====p++p++))+@@@@@@@@@+==±@@@@@@@
@@@+¦p===+@++S+p++@--p+-@+Sp@+¦¦@++-+++-@`@G+¦+@++@¦SpS++@--p+-@@@@@@@@@@@@@@@@@
@
```

6.8.13 Examples: Reading Messages with a MQXMIT format

6.8.13.1 Example 1

Read all messages from the queue.

Windows, Linux or macOS

```
mqbt Read -p MQA1 -q TEST01.Q
```

If the queue had 1 message, it may look like this (the message text begins after 'Raw Data:'):

```
--------------------------- Message # 1 ----------------------------
Transmission Queue Header:
  Remote Q Name:                ABC.XYZ.LQ
  Remote Q Manager Name:        FRED
Message Descriptor:
  Structure ID: MD              Version: 1
*General:
  Message Type: MQMT_DATAGRAM
  Message Priority: 0           Message Persistence: MQPER_NOT_PERSISTENT
  Put Date: 2004-08-30          Expiry Interval: -1
  Put Time: 16:24:20.63         Backout Count: 0
  Reply-To Queue:
  Reply-To Queue Manager:       MQA1
*Report:
  Report: 0
  Feedback Code: MQFB_NONE
*Context:
  User ID:                      'lacrorog    '
  Put-Application Type:         MQAT_WINDOWS_NT
  Put-Application Name:         ':\j2sdk1.4.2_02\bin\java.exe'
  Application Identity Data:    '                                '
  Application Origin Data:      '    '
  Accounting Token:
X'1601051500000075B9755437DA950F828BA62826B2000000000000000000000B'
*Identifiers:
  Message ID                    X'414D51204D5141312020202020202020DA4C334120000901'
  Correlation ID                X'000000000000000000000000000000000000000000000000'
*Attributes:
  Format: MQSTR
  CCSID: 819                    Encoding: 273

Raw Data:
This is a test message.
Line #2
and the last line - #3
```

6.8.14 Examples: Reading Messages with a MQHSAP format

6.8.14.1 Example 1

Read all messages with MQHSAP header from the queue.

Windows, Linux or macOS

```
mqbt Read -p MQA1 -q TEST01.Q
```

If the queue had 1 message, it may look like this (the message text begins after 'Hex Data:'):

```
--------------------------- Message # 1 ----------------------------
SAP Header:
  Structure ID: SAPH            Version: 1
  Header Length: 108            Format: MQSTR
  CCSID: 1200                   Encoding: 273
  Flags: 0                      Client:
  Language:                     Hostname:
  User ID:                      Password:
  System Number: 00
Raw Data:
EDI_DC40                      46C 6422  ABC_PO_DOC01
ABC_PO
                                   SAPIR1    LS  ROMEOINT
                                   SAPIR1    LS  IR1CLNT310
                                   ABC2_PO_HEAD01000
00000100000001L011ROMEOINT
                                   ABC_PO                        DK11
200506CPH-002946-01
                                   00041036681    DKK            DK01070
ABC2_PO_LINE01000                        00000200000102L0117DK4  10020010
1007BLL   010
**********DGASTHODÃˆRNDORF 58                000001100.00   00000000001.00
EA   Z442050820SPACE
```

6.8.15 Examples: Reading Messages with a SMQBAD format

6.8.15.1 Example 1

Read all messages with SMQBAD header from the queue.

Windows, Linux or macOS
```
mqbt Read -p MQA1 -q TEST.BAD.Q
```

If the queue had 1 message, it may look like this (the message text begins after 'Hex Data:'):

```
--------------------------- Message # 1 ----------------------------
SAP Bad Message Header:
  Reason Code: 4109             Reason Text: INVALID_MESSAGE_LENGTH
  Error Type                    1
  CCSID:                        1208
  Encoding:                     273
  Format:                       MQHSAP
  Put-Application Type:         MQAT_JAVA
  Put-Application Name:         'Websphere MQ Client for Java'
  Put Date:                     050408
  Put Time:                     1/16/03

SAP Header:
  Structure ID: SAPH            Version: 1
  Header Length: 108            Format: MQSTR
  CCSID: 1208                   Encoding: 273
  Flags: 0                      Client:
  Language:                     Hostname:
  User ID:                      Password:
  System Number: 00

Raw Data:
EDI_DC40                      46C 6422  ABC_PO_DOC01
ABC_PO
                                    SAPIR1    LS  ROMEOINT
                                    SAPIR1    LS  IR1CLNT310
                                    ABC2_PO_HEAD01000
00000100000001L011ROMEOINT
                                    ABC_PO                              DK11
200506CPH-002946-01
                                    00041036681    DKK                  DK01070
ABC2_PO_LINE01000                         00000200000102L0117DK4  10020010
1007BLL   010
**********DGASTHODÃˆRNDORF 58                        000001100.00   00000000001.00
EA   Z442050820SPACE
```

6.9 Report Message Generation

This section will describe how to generate a report to a PDF, RTF or HTML file from 1 or more messages in a queue. The report function has support for well-known WMQ messages formats: MQMD, MQRFH, MQRFH2, MQCIH, MQDEAD, MQIIH, MQXMIT, MQHSAP and SMQBAD.

6.9.1 Purpose

Use the **Report** command to generate a formatted document containing one of more messages. The formatted can be a PDF, RTF or HTML file.

6.9.2 Syntax

```
mqbt Report [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name -q
Queue_Name [-s Start_Position] [-c Message_Count] [-l layout_format] [-C] [-D]
[-M] [-N] [-H] [-E]
```

6.9.3 Parameters

6.9.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-q	Queue_Name	The name of the queue (Note: Queue names are case sensitive.)

6.9.3.2 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-s	Start_Position	Start position when getting messages from the queue. The default is 1.
-c	Message_Count	Number of messages to be read from the queue. The default is all messages.
-l	layout_format	The layout format of the file: PDF, RTF or HTML
-C		Convert on Get
-D		Remove messages from the queue as they are read (destructive get).
-M		Generate a report that includes the MQMD fields
-N		Generate a report that includes the Named Properties
-H		Generate a report that includes the message data in a hex format.
-E		Generate a report that includes the message data in a hex format but the character convert to EBCDIC.

6.9.4 Examples: Generate a Report with a Raw Data format

6.9.4.1 Example 1
Generate a report using all messages in a queue.

Windows, Linux or macOS
```
mqbt Report -p MQA1 -q TEST01.Q
```

6.9.5 Examples: Generate a Report with a Hex Data format

6.9.5.1 Example 1
Generate a report using all messages in a queue in Hex format.

Windows, Linux or macOS
```
mqbt Report -p MQA1 -q TEST01.Q -H
```

6.9.6 Examples: Generate a Report with a EBCDIC Data format

6.9.6.1 Example 1
Generate a report using all messages in a queue in EBCDIC Hex format

Windows, Linux or macOS
```
mqbt Report -p MQA1 -q TEST01.Q -E
```

6.9.7 Examples: Generate a Report with MQMD format

6.9.7.1 Example 1

Generate a report using all messages in a queue and include the message's MQMD (message descriptor).

Windows, Linux or macOS

```
mqbt Report -p MQA1 -q TEST01.Q -M
```

6.9.7.2 Example 2

Generate a report using all messages in a queue and include the message's MQMD (message descriptor) with Hex Data.

Windows, Linux or macOS

```
mqbt Report -p MQA1 -q TEST01.Q -M -H
```

7 Manipulating Messages of a Topic

This chapter will describe the create messages for an IBM MQ topic.

7.1 Publishing a Message to a Topic

This section will describe how to publish a message to a topic of a queue manager.

7.1.1 Purpose

Use the **Publish** command to insert (write) a messages to a particular topic of a queue manager. The message text can be inputted from the command prompt (shell) or be read from a file. The message can be setup in 1 of 4 ways: binary, string, RFH (Pub/Sub) or RFH2 (JMS).

7.1.2 Syntax

```
mqbt Publish [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name -T
Topic_Name {-f Input_File_Name -t Text_String} [-i mqmd_File_Name] [-S] [-P]
[-R] [-r User_Folder] [-1]
```

7.1.3 Parameters

7.1.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-T	Topic_Name	The name of the topic (Note: Topic names are case sensitive.)

7.1.3.2 Requires at least 1 of the following parameters:

Key	Parameter	Description
-f	Input_File_Name	The full path and filename of the input file.
-t	Text_String	Text string to be used as the message data (enclose text string in quotes).

7.1.3.3 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.

-i	mqmd_File_Name	The full path and filename of the file with the MQMD values
-S		Set the message's MQMD Format field to string (MQSTR).
-P		Set the message's MQMD Format field to RFH (MQRFH version 1).
-R		Set the message's MQMD Format field to RFH2 (MQRFH version 2).
-r	User_Folder	For RFH1 type message, create and set the NameValue pair. For RFH2 type message, create and set the User Folder to the text.
-1		Put each record (line) in the file as a separate message.

7.1.3.4 MQMD IniFile
The description of the MQMD IniFile can be found in Appendix A.

7.1.4 Examples

7.1.4.1 Example 1

Insert a binary message to a topic from a file.

Windows

```
mqbt Publish -p MQA1 -T test/one -f c:\abc.pdf
```

Linux or macOS

```
mqbt Publish -p MQA1 -T test/one -f /abc.pdf
```

7.1.4.2 Example 2

Insert a text string from the command prompt as a 'String' message to a topic.

Windows

```
mqbt Publish -p MQA1 -T test/one -S -t "This is a test message."
```

Linux or macOS

```
mqbt Publish -p MQA1 -T test/one -S -t "This is a test message."
```

7.1.4.3 Example 3

Insert a message from a file in the Publish/Subscribe format (RFH) to a topic.

Windows

```
mqbt Publish -p MQA1 -T test/one -P -r "MQPSCommand RegSub MQPSTopic" -f c:\
myfile.txt
```

Linux or macOS

```
mqbt Publish -p MQA1 -T test/one -P -r "MQPSCommand RegSub MQPSTopic" -f
/myfile.txt
```

7.1.4.4 Example 4

Insert a message from a file in the JMS format (RFH2) to a topic.

Windows

```
mqbt Publish -p MQA1 -T test/one -R -f c:\myfile.txt
```

Linux or macOS

```
mqbt Publish -p MQA1 -T test/one -R -f /myfile.txt
```

7.1.4.5 Example 5

Insert a message from a file in the JMS format (RFH2) with a user folder to a topic.

Windows

```
mqbt Publish -p MQA1 -T test/one -R -f c:\myfile.txt -r
"<usr><up1>ABC</up1></usr>"
```

Linux or macOS

```
mqbt Publish -p MQA1 -T test/one -R -f /myfile.txt -r
"<usr><up1>ABC</up1></usr>"
```

7.2 Importing a File to a Topic

This section will describe how to import a file and create a message that will be written to a topic.

7.2.1 Purpose

Use the **Import** command to write messages to a particular topic from a text file. The message text will be read from a file. The message can be setup in 1 of 4 ways: binary, string, RFH (Pub/Sub) or RFH2 (JMS).

7.2.2 Syntax

```
mqbt Import [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name -q
Topic_Name -f Input_File_Name [-i mqmd_File_Name] [-S] [-P] [-R] [-r
User_Folder]
```

7.2.3 Parameters

7.2.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-q	Topic_Name	The name of the topic (Note: Topic names are case sensitive.)
-f	Input_File_Name	The full path and filename of the input file.

7.2.3.2 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-S		Set the message's MQMD Format field to string (MQSTR).
-P		Set the message's MQMD Format field to RFH (MQRFH version 1).
-R		Set the message's MQMD Format field to RFH2 (MQRFH version 2).
-r	User_Folder	For RFH1 type message, create and set the NameValue pair. For RFH2 type message, create and set the User Folder to the text.

7.2.3.3 MQMD IniFile

The description of the MQMD IniFile can be found in Appendix A.

7.2.4 Examples

7.2.4.1 Example 1

Import a binary message to a topic from a file.

Windows

```
mqbt Import -p MQA1 -T test/one -f c:\abc.pdf
```

Linux or macOS

```
mqbt Import -p MQA1 -T test/one -f /abc.pdf
```

7.2.4.2 Example 2

Import a 'string' message to a topic from a file.

Windows

```
mqbt Import -p MQA1 -T test/one -S -f textfile.txt
```

Linux or macOS

```
mqbt Import -p MQA1 -T test/one -S -f textfile.txt
```

7.2.4.3 Example 3

Import a message from a file in the Publish/Subscribe format (RFH) to a topic.

Windows

```
mqbt Import -p MQA1 -T test/one -P -r "MQPSCommand RegSub MQPSTopic" -f c:\
myfile.txt
```

Linux or macOS

```
mqbt Import -p MQA1 -T test/one -P -r "MQPSCommand RegSub MQPSTopic" -f
/myfile.txt
```

7.2.4.4 Example 4

Import a message from a file in the JMS format (RFH2) to a topic.

Windows

```
mqbt Import -p MQA1 -T test/one -R -f c:\myfile.txt
```

Linux or macOS

```
mqbt Import -p MQA1 -T test/one -R -f /myfile.txt
```

7.2.4.5 Example 5

Import a message from a file in the JMS format (RFH2) with a user folder to a topic.

Windows

```
mqbt Import -p MQA1 -T test/one -R -f c:\myfile.txt -r
"<usr><up1>ABC</up1></usr>"
```

Linux or macOS

```
mqbt Import -p MQA1 -T test/one -R -f /myfile.txt -r
"<usr><up1>ABC</up1></usr>"
```

7.3 Export a Message from a Topic

This section will describe how to export a message to a file from a topic.

7.3.1 Purpose

Use the **Export** command to write messages from a topic to a text file. Reading the messages in the topic can be done either destructively or non-destructively. Also, the 'Convert on Get' option can be specified for the MQGET.

7.3.2 Syntax

```
mqbt Export [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name -q
Topic_Name -f Output_File_Name [-s Start_Position] [-t Trailer_Text] [-c
Message_Count] [-D] [-A] [-X] [-1]
```

7.3.3 Parameters

7.3.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-q	Topic_Name	The name of the topic (Note: Topic names are case sensitive.)
-f	Output_File_Name	The full path and filename of the output file.

7.3.3.2 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-s	Start_Position	Start position when getting messages from the topic. The default is 1.
-c	Message_Count	Number of messages to be read from the topic. The default is all messages.
-t	Trailer_Text	Trailer text string addended after each exported message. Use "\n" to represent CRLF on Windows and LF on Linux or macOS/Linux.
-C		Convert on Get
-D		Remove messages from the topic as they are read (destructive get).
-A		If this parameter is present then the messages will be appended to output file.
-X		Remove any MQ header during the export process
-1		Write all messages to same file.

7.3.4 Examples

7.3.4.1 Example 1
Export all messages in the 'TEST01.Q' topic.

Windows
```
mqbt Export -p MQA1 -T test/one -f textfile.txt
```

Linux or macOS
```
mqbt Export -p MQA1 -T test/one -f textfile.txt
```

If there were 8 messages in the topic, then the output files would look like:

```
textfile_0001.txt
textfile_0002.txt
textfile_0003.txt
textfile_0004.txt
textfile_0005.txt
textfile_0006.txt
textfile_0007.txt
textfile_0008.txt
```

7.3.4.2 Example 2
Export 10 messages starting at message number 25 from the 'TEST01.Q' topic and delete them from the topic.

Windows
```
mqbt Export -p MQA1 -T test/one -f textfile.txt -s 25 -c 25 -D
```

Linux or macOS
```
mqbt Export -p MQA1 -T test/one -f textfile.txt -s 25 -c 25 -D
```

7.3.4.3 Example 3
Export all messages doing a destructive get (deleting the messages) with the option of 'Convert on Get' from the 'TEST01.Q' topic.

Windows
```
mqbt Export -p MQA1 -T test/one -f textfile.txt -D -C
```

Linux or macOS
```
mqbt Export -p MQA1 -T test/one -f textfile.txt -D -C
```

7.3.4.4 Example 4
Export 10 messages from the 'TEST01.Q' topic, appending the messages to a file, appending just a NewLine (CRLF or LF) and delete them from the topic.

Windows
```
mqbt Export -p MQA1 -T test/one -f textfile.txt -t "\n" -A -D
```

Linux or macOS
```
mqbt Export -p MQA1 -T test/one -f textfile.txt -t "\n" -A -D
```

7.3.4.5 Example 5

Export 10 messages from the 'TEST01.Q' topic, appending the messages to a file, appending trailer text and delete them from the topic.

Windows

```
mqbt Export -p MQA1 -T test/one -f textfile.txt -t "\n\n" -A -D
```

Linux or macOS

```
mqbt Export -p MQA1 -T test/one -f textfile.txt -t "\n\n" -A -D
```

7.4 Subscribe to a Topic

This section will describe how to scribe to a topic to receive a message. The subscribe function has support for well-known WMQ messages formats: MQMD, MQRFH, MQRFH2, MQCIH, MQDEAD, MQIIH, MQXMIT, MQHSAP and SMQBAD.

7.4.1 Purpose

Use the **Subscribe** command to receive messages of a topic.

7.4.2 Syntax

```
mqbt Subscribe [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name -T
Topic_Name [-s Start_Position] [-c Message_Count] [-C] [-M] [-H] [-E]
```

7.4.3 Parameters

7.4.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-T	Topic_Name	The name of the topic (Note: Topic names are case sensitive.)

7.4.3.2 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-s	Start_Position	Start position when getting messages from the topic. The default is 1.
-c	Message_Count	Number of messages to be read from the topic. The default is all messages.
-C		Convert on Get
-M		Display the MQMD fields before display the message's data.
-H		Display the message data in a hex format.
-E		Display the message data in a hex format but the character convert to EBCDIC.

7.4.4 Examples: Reading Messages with a Raw Data format

7.4.4.1 Example 1

Subscribe to a topic and receive messages.

Windows, Linux or macOS

```
mqbt Subscribe -p MQA1 -T test/one
```

If the topic had 1 message, it may look like this (the message text begins after 'Raw Data:'):

```
--------------------------- Message # 1 ----------------------------
Raw Data:
This is a test message.
Line #2
and the last line - #3
```

7.4.5 Examples: Reading Messages with a Hex Data format

7.4.5.1 Example 1

Read all messages in Hex format from the topic.

Windows, Linux or macOS

```
mqbt Subscribe -p MQA1 -T test/one -H
```

If the topic had 1 message, it may look like this (the message text begins after 'Hex Data:'):

```
--------------------------- Message # 1 ----------------------------
Hex Data:
  00000000  54 68 69 73 20 69 73 20 61 20 74 65 73 74 20 6D  'This is a test m'
  00000010  65 73 73 61 67 65 2E 0A 4C 69 6E 65 20 23 32 0A  'essage. Line #2 '
  00000020  61 6E 64 20 74 68 65 20 6C 61 73 74 20 6C 69 6E  'and the last lin'
  00000030  65 20 2D 20 23 33 0A                             'e - #3          '
```

7.4.6 Examples: Reading Messages with a EBCDIC Data format

7.4.6.1 Example 1

Read all messages in EBCDIC Hex format from the topic.

Windows, Linux or macOS

```
mqbt Subscribe -p MQA1 -T test/one -E
```

If the topic had 1 message, it may look like this (the message text begins after 'EBCDIC Data:'):

```
--------------------------- Message # 1 ----------------------------
EBCDIC Data:
  00000000  2E 2E 2E 2E 2E 2E 2E 2E 2F 2E 2E 2E 2E 2E 2E 5F  '......../......_'
  00000010  2E 2E 2E 2F 2E 2E 2E 2E 3C 2E 3E 2E 2E 2E 2E 2E  '.../....<.>.....'
  00000020  2F 3E 2E 2E 2E 2E 2E 2E 25 2F 2E 2E 2E 25 2E 3E  '/>......%/...%.>'
  00000030  2E 2E 2E 2E 2E 2E 2E                             '.......         '
```

7.4.7 Examples: Reading Messages with a MQMD format

7.4.7.1 Example 1

Read all messages from the topic and display the message's MQMD (message descriptor).

Windows, Linux or macOS

```
mqbt Subscribe -p MQA1 -T test/one -M
```

If the topic had 1 message, it may look like this (the message text begins after 'Raw Data:'):

```
--------------------------- Message # 1 ----------------------------
Message Descriptor:
*General:
  Version: 2                   Message Type: MQMT_DATAGRAM
  Message Priority: 0          Message Persistence: MQPER_NOT_PERSISTENT
  Put Date: 2004-08-30         Expiry Interval: -1
  Put Time: 01:40:12.280       Backout Count: 0
  Reply-To Topic:
  Reply-To Topic Manager:      MQA1
*Report:
  Report: 0                    Original Length: -1
  Feedback Code: MQFB_NONE
*Context:
  User ID:                     'rlacroix    '
  Put-Application Type:        MQAT_WINDOWS_NT
  Put-Application Name:        'C:\WINNT\system32\javaw.exe '
  Application Identity Data:   '                                '
  Application Origin Data:     '    '
  Accounting Token:
X'16010515000000235F636B54EC114F828BA628E803000000000000000000000B'
*Identifiers:
  Message ID:
X'414D51204D5141312020202020202020F9BB324120000401'
  Correlation ID:
X'000000000000000000000000000000000000000000000000'
  Group ID:
X'000000000000000000000000000000000000000000000000'
*Segmentation:
  Logical Sequence Number: 1
  Offset: 1                    Message Flags: 0
*Attributes:
  Format: MQSTR                Message Data Length: 55
  CCSID: 819                   Encoding: 273

Raw Data:
This is a test message.
Line #2
and the last line - #3
```

7.4.7.2 Example 2

Read all messages from the topic and display the message's MQMD (message descriptor) with Hex Data.

Windows, Linux or macOS

```
mqbt Subscribe -p MQA1 -T test/one -M -H
```

If the topic had 1 message, it may look like this (the message text begins after 'Hex Data:'):

```
--------------------------- Message # 1 ----------------------------
Message Descriptor:
*General:
  Version: 2                     Message Type: MQMT_DATAGRAM
  Message Priority: 0            Message Persistence: MQPER_NOT_PERSISTENT
  Put Date: 2004-08-30           Expiry Interval: -1
  Put Time: 01:40:12.280         Backout Count: 0
  Reply-To Topic:
  Reply-To Topic Manager:        MQA1
*Report:
  Report: 0                      Original Length: -1
  Feedback Code: MQFB_NONE
*Context:
  User ID:                       'rlacroix    '
  Put-Application Type:          MQAT_WINDOWS_NT
  Put-Application Name:          'C:\WINNT\system32\javaw.exe '
  Application Identity Data:     '                                '
  Application Origin Data:       '    '
  Accounting Token:
X'16010515000000235F636B54EC114F828BA628E80300000000000000000000000B'
*Identifiers:
  Message ID:                    X'414D51204D51413120202020202020F9BB324120000401'
  Correlation ID:                X'000000000000000000000000000000000000000000000000'
  Group ID:                      X'000000000000000000000000000000000000000000000000'
*Segmentation:
  Logical Sequence Number: 1
  Offset: 1                      Message Flags: 0
*Attributes:
  Format: MQSTR                  Message Data Length: 55
  CCSID: 819                     Encoding: 273

Hex Data:
  00000000  54 68 69 73 20 69 73 20 61 20 74 65 73 74 20 6D  'This is a test m'
  00000010  65 73 73 61 67 65 2E 0A 4C 69 6E 65 20 23 32 0A  'essage. Line #2 '
  00000020  61 6E 64 20 74 68 65 20 6C 61 73 74 20 6C 69 6E  'and the last lin'
  00000030  65 20 2D 20 23 33 0A                             'e - #3          '
```

7.4.8 Examples: Reading Messages with a MQRFH format

7.4.8.1 Example 1

Read all messages from the topic.

Windows, Linux or macOS

```
mqbt Subscribe -p MQA1 -T test/one
```

If the topic had 1 message, it may look like this (the message text begins after 'Raw Data:'):

```
--------------------------- Message # 1 ----------------------------
Rules and Formatting Header v1:
  Structure ID: RFH            Version: 1
  Header Length: 134           Format: MQSTR
  CCSID: 819                   Encoding: 273
  Flags: 0
Raw Data:
MQPSCommand  RegSub  MQPSTopic topic MQPSQName subscribertopic MQPSStreamName
stream MQPSQMgrName MQA1
```

7.4.9 Examples: Reading Messages with a MQRFH2 format

7.4.9.1 Example 1

Read all messages from the topic.

Windows, Linux or macOS
```
mqbt Subscribe -p MQA1 -T test/one -H
```

If the topic had 1 message, it may look like this (the message text begins after 'Raw Data:'):

```
--------------------------- Message # 1 ----------------------------
Rules and Formatting Header v2:
  Structure ID: RFH             Version: 2
  Header Length: 152            Format: MQSTR
  CCSID: 437                    Encoding: 546
  Name Value CCSID: 1208        Flags: 0
Folder:
<mcd><Msd>jms_text</Msd></mcd>
Folder:
<jms><jmsp111>oneone</jmsp111></jms>
Folder:
<usr><u111>oneoneoneone</u111></usr>
Raw Data:
this is the message body text
```

7.4.10 Examples: Reading Messages with a MQDEAD format

7.4.10.1 Example 1

Read all messages with MQDEAD header from the topic.

Windows, Linux or macOS

```
mqbt Subscribe -p MQA1 -T test/one
```

If the topic had 1 message, it may look like this (the message text begins after 'Raw Data:'):

```
--------------------------- Message # 1 ----------------------------
Dead Letter Header:
  Reason Code: 265             Reason Text: MQFB_APPL_CANNOT_BE_STARTED
  Destination Q Name:          SYSTEM.DEFAULT.INITIATION.QUEUE
  Destination QMgr Name:       MQA1
  CCSID:                       437
  Encoding:                    546
  Format:                      MQTRIG
  Put-Application Type:        MQAT_WINDOWS_NT
  Put-Application Name:        'RUNMQTRM                    '
  Put Date:                    2004-08-30
  Put Time:                    12:24:56.050

Raw Data:
TM      BAD_TRIG_QUEUE                                  BAD_PROC

            c:\GoToNothing.bat
```

7.4.11 Examples: Reading Messages with a MQCIH format

7.4.11.1 Example 1

Read all messages with MQCIH header from the topic.

Windows, Linux or macOS

```
mqbt Subscribe -p MQA1 -T test/one
```

If the topic had 1 message, it may look like this (the message text begins after 'Raw Data:'):

```
--------------------------- Message # 1 ----------------------------
CICS Information Header:
  Structure ID: CIH              Version: 1
  Header Length: 164             Format:
  CCSID: 819                     Encoding: 1
  Flags: 0                       Return Code: 0
  Comp Code: 0                   Reason: 0
  UOW Control: 273               Get Wait Interval: -2
  Link Type: 1                   Output Data Length: -1
  Facility Keep Time: 0          ADS Descriptor: 0
  Conversational Task: 0         Task End Status: 0
  Function:                      Facility: X'0000000000000000'
  Abend Code:                    Authenticator:
  Reply To Format:               Remote SysId:
  Remote TransId:                TransactionId:
  Facility Like:                 Attention Id:
  Start Code:                    Cancel Code:
  Next TransactionId:            Cursor Position: 0
  Error Offset: 0                Input Item: 0
Raw Data:
```

7.4.12 Examples: Reading Messages with a MQIIH format

7.4.12.1 Example 1

Read all messages with MQIIH header from the topic.

Windows, Linux or macOS

```
mqbt Subscribe -p MQA1 -T test/one
```

If the topic had 1 message, it may look like this (the message text begins after 'Raw Data:'):

```

--------------------------- Message # 1 ----------------------------
IMS Information Header:
  Structure ID: IIH             Version: 1
  Header Length: 84             Format:
  CCSID: 0                      Encoding: 785
  Flags: 0                      LTerm Override:
  MFS Map Name:                 Reply To Format:
  Authenticator:                Tran State: X'31'
  Commit Mode: X'20'            Security Scope: X'A5'
  Tran InstanceId: X'BA28F0725B3FB1614040404040404040'
Raw Data:
+¦¦=@@@@@+¦¦=±+-=··±==)======··±±=±±±=)=˜++++@=====p++p++))+@@@@@@@@@+==±@@@@@@@
@@@+¦p===+@++S+p++@--p+-@+Sp@+¦¦@++-+++-@`@G+¦+@++@¦SpS++@--p+-@@@@@@@@@@@@@@@@@@
@
```

7.4.13 Examples: Reading Messages with a MQXMIT format

7.4.13.1 Example 1

Read all messages from the topic.

Windows, Linux or macOS

```
mqbt Subscribe -p MQA1 -T test/one
```

If the topic had 1 message, it may look like this (the message text begins after 'Raw Data:'):

```
--------------------------- Message # 1 ----------------------------
Transmission Topic Header:
  Remote Q Name:               ABC.XYZ.LQ
  Remote Q Manager Name:       FRED
Message Descriptor:
  Structure ID: MD             Version: 1
*General:
  Message Type: MQMT_DATAGRAM
  Message Priority: 0          Message Persistence: MQPER_NOT_PERSISTENT
  Put Date: 2004-08-30         Expiry Interval: -1
  Put Time: 16:24:20.63        Backout Count: 0
  Reply-To Topic:
  Reply-To Topic Manager:      MQA1
*Report:
  Report: 0
  Feedback Code: MQFB_NONE
*Context:
  User ID:                     'lacrorog    '
  Put-Application Type:        MQAT_WINDOWS_NT
  Put-Application Name:        ':\j2sdk1.4.2_02\bin\java.exe'
  Application Identity Data:   '                                '
  Application Origin Data:     '    '
  Accounting Token:
X'1601051500000075B9755437DA950F828BA62826B2000000000000000000000B'
*Identifiers:
  Message ID                   X'414D51204D5141312020202020202020DA4C334120000901'
  Correlation ID               X'000000000000000000000000000000000000000000000000'
*Attributes:
  Format: MQSTR
  CCSID: 819                   Encoding: 273

Raw Data:
This is a test message.
Line #2
and the last line - #3
```

7.4.14 Examples: Reading Messages with a MQHSAP format

7.4.14.1 Example 1

Read all messages with MQHSAP header from the topic.

Windows, Linux or macOS

```
mqbt Subscribe -p MQA1 -T test/one
```

If the topic had 1 message, it may look like this (the message text begins after 'Hex Data:'):

```
--------------------------- Message # 1 ----------------------------
SAP Header:
  Structure ID: SAPH             Version: 1
  Header Length: 108             Format: MQSTR
  CCSID: 1200                    Encoding: 273
  Flags: 0                       Client:
  Language:                      Hostname:
  User ID:                       Password:
  System Number: 00
Raw Data:
EDI_DC40                     46C 6422  ABC_PO_DOC01
ABC_PO
                                   SAPIR1    LS  ROMEOINT
                                   SAPIR1    LS  IR1CLNT310
                                   ABC2_PO_HEAD01000
00000100000001L011ROMEOINT
                                   ABC_PO                            DK11
200506CPH-002946-01
                                   00041036681    DKK                DK01070
ABC2_PO_LINE01000                        00000200000102L0117DK4  10020010
1007BLL   010
**********DGASTHODÃˆRNDORF 58                        000001100.00   00000000001.00
EA   Z442050820SPACE
```

7.4.15 Examples: Reading Messages with a SMQBAD format

7.4.15.1 Example 1

Read all messages with SMQBAD header from the topic.

Windows, Linux or macOS

```
mqbt Subscribe -p MQA1 -T test/bad/one
```

If the topic had 1 message, it may look like this (the message text begins after 'Hex Data:'):

```
--------------------------- Message # 1 ----------------------------
SAP Bad Message Header:
  Reason Code: 4109             Reason Text: INVALID_MESSAGE_LENGTH
  Error Type                    1
  CCSID:                        1208
  Encoding:                     273
  Format:                       MQHSAP
  Put-Application Type:         MQAT_JAVA
  Put-Application Name:         'Websphere MQ Client for Java'
  Put Date:                     050408
  Put Time:                     1/16/03

SAP Header:
  Structure ID: SAPH            Version: 1
  Header Length: 108            Format: MQSTR
  CCSID: 1208                   Encoding: 273
  Flags: 0                      Client:
  Language:                     Hostname:
  User ID:                      Password:
  System Number: 00

Raw Data:
EDI_DC40                      46C 6422  ABC_PO_DOC01
ABC_PO
                                  SAPIR1    LS  ROMEOINT
                                  SAPIR1    LS  IR1CLNT310
                                  ABC2_PO_HEAD01000
00000100000001L011ROMEOINT
                                  ABC_PO                              DK11
200506CPH-002946-01
                                  00041036681    DKK                  DK01070
ABC2_PO_LINE01000                       00000200000102L0117DK4  10020010
1007BLL   010
**********DGASTHODÃˆRNDORF 58                     000001100.00   00000000001.00
EA   Z442050820SPACE
```

7.5 Report Message Generation for a Topic

This section will describe how to generate a report to a PDF, RTF or HTML file from 1 or more messages in a topic. The report function has support for well-known WMQ messages formats: MQMD, MQRFH, MQRFH2, MQCIH, MQDEAD, MQIIH, MQXMIT, MQHSAP and SMQBAD.

7.5.1 Purpose

Use the **Report** command to generate a formatted document containing one of more messages. The formatted can be a PDF, RTF or HTML file.

7.5.2 Syntax

```
mqbt Report [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name -T
Topic_Name [-s Start_Position] [-c Message_Count] [-l layout_format] [-C] [-M]
[-N] [-H] [-E]
```

7.5.3 Parameters

7.5.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-T	Topic_Name	The name of the topic (Note: Topic names are case sensitive.)

7.5.3.2 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-s	Start_Position	Start position when getting messages from the topic. The default is 1.
-c	Message_Count	Number of messages to be read from the topic. The default is all messages.
-l	layout_format	The layout format of the file: PDF, RTF or HTML
-C		Convert on Get
-M		Generate a report that includes the MQMD fields
-N		Generate a report that includes the named properties
-H		Generate a report that includes the message data in a hex format.
-E		Generate a report that includes the message data in a hex format but the character convert to EBCDIC.

7.5.4 Examples: Generate a Report with a Raw Data format

7.5.4.1 Example 1
Generate a report using all messages in a topic.

Windows, Linux or macOS
```
mqbt Report -p MQA1 -T test/one
```

7.5.5 Examples: Generate a Report with a Hex Data format

7.5.5.1 Example 1
Generate a report using all messages in a topic in Hex format.

Windows, Linux or macOS
```
mqbt Report -p MQA1 -T test/one -H
```

7.5.6 Examples: Generate a Report with a EBCDIC Data format

7.5.6.1 Example 1
Generate a report using all messages in a topic in EBCDIC Hex format

Windows, Linux or macOS
```
mqbt Report -p MQA1 -T test/one -E
```

7.5.7 Examples: Generate a Report with MQMD format

7.5.7.1 Example 1
Generate a report using all messages in a topic and include the message's MQMD (message descriptor).

Windows, Linux or macOS
```
mqbt Report -p MQA1 -T test/one -M
```

7.5.7.2 Example 2
Generate a report using all messages in a topic and include the message's MQMD (message descriptor) with Hex Data.

Windows, Linux or macOS
```
mqbt Report -p MQA1 -T test/one -M -H
```

8 Email

This chapter will describe the process of sending and receiving messages between an IBM MQ queue and an email system.

8.1 Get an Email Message and Put an MQ Message

This section will describe how to retrieve one or more email messages (via POP) and put each email message as an MQ message to an MQ queue.

8.1.1 Purpose

Use the **GetEmail** command to retrieve one or more email messages (via POP) and put each email message as an MQ message to an MQ queue. The message can be setup in 1 of 4 ways: binary, string, RFH (Pub/Sub) or RFH2 (JMS).

8.1.2 Syntax

```
mqbt GetEmail [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name -q
Queue_Name -f Input_File_Name -e Email_File_Name [-i mqmd_File_Name] [-S] [-P]
[-R] [-r User_Folder]
```

8.1.3 Parameters

8.1.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-q	Queue_Name	The name of the queue (Note: Queue names are case sensitive.)
-e	Email_File_Name	The full path and filename of the file with the POP email configuration

8.1.3.2 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-i	mqmd_File_Name	The full path and filename of the file with the MQMD values
-S		Set the message's MQMD Format field to string (MQSTR).
-P		Set the message's MQMD Format field to RFH (MQRFH version 1).
-R		Set the message's MQMD Format field to RFH2 (MQRFH version 2).
-r	User_Folder	For RFH1 type message, create and set the NameValue pair. For RFH2 type message, create and set the User Folder to the text.

8.1.3.3 Email IniFile

The email IniFile has 7 keywords in the POP stanza.

- **email.host** keyword specifies the email system hostname or IP address
- **email.port** keyword (optional) specifies the email system's port number (default is 110)
- **email.ssl** keyword specifies if SSL should be used or not (true/false)
- **email.auth** keyword specifies if authentication should be used or not (true/false)
- **email.user** keyword specifies the UserID/email address to be used when logging in
- **email.pwd** keyword specifies the Password to be used when logging in
- **email.delete** keyword specifies if the email message should be deleted (true/false)

Sample email_pop.ini file

```
[POP]
email.host=mail.acme.com
email.port=995
email.ssl=true
email.auth=true
email.user=tester@acme.com
email.pwd=barney
email.delete=false
```

8.1.4 Examples

8.1.4.1 Example 1

Retrieve an email message and put it to a queue as a binary MQ message.

Windows, Linux or macOS
```
mqbt GetEmail -p MQA1 -q TEST01.Q -e email_pop.ini
```

8.1.4.2 Example 2

Retrieve an email message and put it to a queue as a 'string' MQ message.

Windows, Linux or macOS
```
mqbt GetEmail -p MQA1 -q TEST01.Q -S -e email_pop.ini
```

8.1.4.3 Example 3

Retrieve an email message and put it to a queue as a Publish/Subscribe format (RFH) MQ message.

Windows, Linux or macOS
```
mqbt GetEmail -p MQA1 -q TEST01.Q -P -r "MQPSCommand RegSub MQPSTopic" -e
email_pop.ini
```

8.1.4.4 Example 4

Retrieve an email message and put it to a queue as a JMS format (RFH2) MQ message.

Windows, Linux or macOS
```
mqbt GetEmail -p MQA1 -q TEST01.Q -R -e email_pop.ini
```

8.1.4.5 Example 5

Retrieve an email message and put it to a queue as a JMS format (RFH2) with a user folder MQ message.

Windows, Linux or macOS
```
mqbt GetEmail -p MQA1 -q TEST01.Q -R -e email_pop.ini -r
"<usr><up1>ABC</up1></usr>"
```

8.2 Get an MQ Message and Send an Email Message

This section will describe how to get one or more MQ messages from an MQ queue and send each MQ message as an email message (via SMTP).

8.2.1 Purpose

Use the **SendEmail** command to get one or more MQ messages from an MQ queue and send each MQ message as an email message (via SMTP). Reading the messages in the queue can be done either destructively or non-destructively. Also, the 'Convert on Get' option can be specified for the MQGET.

8.2.2 Syntax

```
mqbt SendEmail [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name -q
Queue_Name -e Email_File_Name [-s Start_Position] [-t Trailer_Text] [-c
Message_Count] [-D] [-X] [-1]
```

8.2.3 Parameters

8.2.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-q	Queue_Name	The name of the queue (Note: Queue names are case sensitive.)
-e	Email_File_Name	The full path and filename of the file with the SMTP email configuration

8.2.3.2 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-s	Start_Position	Start position when getting messages from the queue. The default is 1.
-c	Message_Count	Number of messages to be read from the queue. The default is all messages.
-t	Trailer_Text	Trailer text string addended after each exported message. Use "\n" to represent CRLF on Windows and LF on Linux or macOS/Linux.
-C		Convert on Get
-D		Remove messages from the queue as they are read (destructive get).
-X		Remove any MQ header during the export process
-1		Write all messages to same file.

8.2.3.3 Email IniFile

The email IniFile has 9 keywords in the SMTP stanza.

- **email.host** keyword specifies the email system hostname or IP address
- **email.port** keyword (optional) specifies the email system's port number (default is 465)
- **email.ssl** keyword specifies if SSL should be used or not (true/false)
- **email.auth** keyword specifies if authentication should be used or not (true/false)
- **email.user** keyword specifies the UserID/email address to be used when logging in
- **email.pwd** keyword specifies the Password to be used when logging in
- **email.from** keyword specifies if the from email address to be used for the email message
- **email.to** keyword specifies 1 or more email addresses to be used for the email message
- **email.subject** keyword specifies the subject text of the email message

Sample email_smtp.ini file

```
[SMTP]
email.host=mail.acme.com
email.port=465
email.ssl=true
email.auth=true
email.user=tester@acme.com
email.pwd=barney
email.from=tester@acme.com
email.to=receiver@acme.com
email.subject=message from mq queue
```

8.2.4 Examples

8.2.4.1 Example 1
Send all MQ messages in the 'TEST01.Q' queue as email messages.

Windows, Linux or macOS
```
mqbt SendEmail -p MQA1 -q TEST01.Q -e email_smtp.ini
```

8.2.4.2 Example 2
Send 10 MQ messages starting at MQ message number 25 in the 'TEST01.Q' queue as email messages.

Windows, Linux or macOS
```
mqbt SendEmail -p MQA1 -q TEST01.Q -e email_smtp.ini -s 25 -c 25 -D
```

8.2.4.3 Example 3
Send all messages doing a destructive get (deleting the messages) with the option of 'Convert on Get' in the 'TEST01.Q' queue as email messages.

Windows, Linux or macOS
```
mqbt SendEmail -p MQA1 -q TEST01.Q -e email_smtp.ini -D -C
```

8.2.4.4 Example 4
Send 10 messages (read and delete) from the 'TEST01.Q' queue, appending just a NewLine (CRLF or LF) to the email messages.

Windows, Linux or macOS
```
mqbt SendEmail -p MQA1 -q TEST01.Q -e email_smtp.ini -t "\n" -D
```

8.2.4.5 Example 5
Send 10 messages (read and delete) from the 'TEST01.Q' queue, appending a simple text message to the email messages.

Windows, Linux or macOS
```
mqbt SendEmail -p MQA1 -q TEST01.Q -e email_smtp.ini -t "\nThe End\n" -D
```

9 MQ Tools

This chapter will describes how to invoke and use the MQ Tools included with MQ Batch Toolkit.

9.1 CheckUp

This section will describe how to invoke and use the CheckUp tool. The CheckUp function will verify that attributes of MQ objects are valid and exists. CheckUp scans queues, channels, namelists, processes and queue manager attributes.

CheckUp has roughly 20 rules that are verified against the queue manager's objects. i.e. Does the BaseQueue name of an Alias Queue point to a valid local or remote queue.

Anytime CheckUp finds an issue or potential problem, the information is written to a report file. The file name is ***MQBT_CheckUp_Report.txt*** and it is written to the current directory. The output will include the queue manager name, MQ version, channel name, hostname or an IP Address and port number.

9.1.1 Purpose

Use the **CheckUp** command to check / validate the attributes of MQ objects.

9.1.2 Syntax

```
mqbt CheckUp [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name
```

9.1.3 Parameters

9.1.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.

9.1.3.2 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.

9.1.4 Examples

9.1.4.1 Example 1

Perform a checkup of queue manager MQA1.

Windows, Linux or macOS

```
mqbt CheckUp -p MQA1
```

```
Channel=APP.CH01 has a blank MCAUSER and no security exit or SSLCipher. This is a
security risk.
Channel=MY.TEST.EXIT.I has a blank MCAUSER and no security exit or SSLCipher. This is
a security risk.
Channel=SYSTEM.ADMIN.SVRCONN has a blank MCAUSER and no security exit or SSLCipher.
This is a security risk.
Channel=SYSTEM.AUTO.SVRCONN has a blank MCAUSER and no security exit or SSLCipher.
This is a security risk.
Channel=TEST.CHL has a blank MCAUSER and no security exit or SSLCipher. This is a
security risk.
Channel=TEST.EXIT has a blank MCAUSER and no security exit or SSLCipher. This is a
security risk.
Channel=TEST.NOEXIT has a blank MCAUSER and no security exit or SSLCipher. This is a
security risk.
Queue=MQWT2.XMIT is Get Inhibited.
Queue=CSQ6.XMIT has USAGE(XMITQ) with triggering set on but INITQ is blank.
Queue=CSQ6.XMIT has USAGE(XMITQ) with triggering set on but TRIGDATA is blank.
Queue=MQA1.XMIT has triggering set off but it has a INITQ defined
(SYSTEM.CHANNEL.INITQ).
Queue=MQWT1.XMIT has USAGE(XMITQ) with triggering set on but TRIGDATA is blank.
Queue=MQWT2.XMIT has triggering set off but it has a INITQ defined
(SYSTEM.CHANNEL.INITQ).
Queue=TEST.MQBT.TRIG : INITQ=SYSTEM.DEFAULT.INITIATION.QUEUE does not have a trigger
monitor running against it.
Queue=WMQA.XMIT has USAGE(XMITQ) with triggering set on but TRIGDATA is blank.
```

9.2 PortScan

This section will describe how to invoke and use the PortScan tool. PortScan tool will scan a range of ports for a given server looking for a queue manager's MCA, using the standard (system default) channel names, in order to make a successful connection.

When a user inputs a hostname or an IP Address, PortScan will scan a range of ports looking for a queue manager's MCA, using the standard (system default) channel names, in order to make a successful connection.

PortScan has the ability to search across a range of IP addresses and scan the port range for each IP address.

Anytime PortScan successfully connects to a queue manager, the information is written to a CSV (Comma-Separated-Value) file. The output will include the queue manager name, MQ version, channel name, hostname or an IP Address and port number.

9.2.1 Purpose

Use the **PortScan** command to remove the messages of a particular queue of a queue manager.

9.2.2 Syntax

```
mqbt PortScan -h hostname [-s start_port_number] [-e end_port_number]
```

9.2.3 Parameters

9.2.3.1 Required Parameters

Key	Parameter	Description
-h	hostname	Hostname or IP address of the remote server

9.2.3.2 Optional Parameters

Key	Parameter	Description
-s	start_port_number	Starting port number
-e	end_port_number	Ending port number

9.2.4 Examples

9.2.4.1 Example 1
Scan ports 1414 to 10000 on host 10.10.10.10

Windows, Linux or macOS
```
mqbt PortScan -h 10.10.10.10
```

9.2.4.2 Example 2
Scan ports 1400 to 2000 on host server01

Windows, Linux or macOS
```
mqbt PortScan -h server01 -s 1400 -e 2000
```

9.2.4.3 Example 3
Scan ports 1400 to 5000 on all IP addresses 10.10.1.1 to 10.10.255.255 (65536 servers)

Windows, Linux or macOS
```
mqbt PortScan -h 10.10.*.* -s 1400 -e 5000
```

10 Stress Testing Tools

This chapter will describes how to invoke and use the Stress Testing Tools included with MQ Batch Toolkit.

10.1 Get Server

This section will describe how to invoke and use the Get Server Stress Testing tool. The Get Server will continuously consume messages from a queue and show statistics of its current state.

For the Stress Testing Servers (Get, Put, SIM Client & SIM Server), you can have unlimited number of them running (up to what your box can support). Also, while any of the Servers are running, the user can still use MQ Batch Toolkit.

10.1.1 Purpose

Use the **GetServer** command to remove the messages of a particular queue of a queue manager.

10.1.2 Syntax

```
mqbt GetServer [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name -q
Queue_Name {-d Days -h Hours -m Minutes -n Max_Message_Count}
```

10.1.3 Parameters

10.1.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-q	Queue_Name	The name of the queue (Note: Queue names are case sensitive.)

10.1.3.2 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-d	Days	Consumes messages for 'x' days.
-h	Hours	Consumes messages for 'x' hours.
-m	Hours	Consumes messages for 'x' minutes.
-n	Max_Message_Count	Maximum number of messages to be read from the queue before terminating.

10.1.4 Examples

10.1.4.1 Example 1

Consume the messages from a queue of a queue manager for 4 hours.

Windows, Linux or macOS

```
mqbt GetServer -p MQA1 -q TEST01.Q -h 4
```

10.1.4.2 Example 2

Consume the messages from a queue of a queue manager for 8 days, 4 hours and 35 minutes.

Windows, Linux or macOS

```
mqbt GetServer -p MQA1 -q TEST01.Q -d 8 -h 4 -m 35
```

10.1.4.3 Example 3

Consume the messages from a queue of a queue manager until 5000 messages have been consumed.

Windows, Linux or macOS

```
mqbt GetServer -p MQA1 -q TEST01.Q -n 5000
```

10.2 Put Server

This section will describe how to invoke and use the Put Server Stress Testing tool. The Put Server will put messages to a queue. It can be used to stress test a 'Server Component'. The user can control the feed by putting a delay between the MQPUTs. The messages data can be from a simple input text or from a plain text file or from a Backup file (this file can contain many messages with MQMDs). A Backup file can be either a SQLite Database (*.mqsdb) or a VEQ formatted file (*.veq).

For the Stress Testing Servers (Get, Put, SIM Client & SIM Server), you can have unlimited number of them running (up to what your box can support). Also, while any of the Servers are running, the user can still use MQ Batch Toolkit.

10.2.1 Purpose

Use the **PutServer** command to write (put) a messages to a particular queue of a queue manager. The message text can be inputted from the command prompt (shell) or be read from a file. The message can be setup in 1 of 4 ways: binary, string, RFH (Pub/Sub) or RFH2 (JMS).

10.2.2 Syntax

```
mqbt PutServer -a Path_FileName_for_CommProfileDB -p profileName -q QueueName
-n NumberOfMessages {-f Path_FileName -t text_message} [-i mqmd_File_Name] [-
S] [-P] [-R] [-r usrFolderData] [-d PutDelayInMS]
```

10.2.3 Parameters

10.2.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-q	Queue_Name	The name of the queue (Note: Queue names are case sensitive.)
-n	Number_of_Messages	Maximum number of messages to be read from the queue before terminating.

10.2.3.2 Requires at least 1 of the following parameters:

Key	Parameter	Description
-f	Path_File_Name	The full path and filename of the input file.
-t	Text_Message	Text string to be used as the message data (enclose text string in quotes).

10.2.3.3 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-i	mqmd_File_Name	The full path and filename of the file with the MQMD values
-d	Put_Delay	Delay between Puts in milliseconds (ms).
-S		Set the message's MQMD Format field to string (MQSTR).
-P		Set the message's MQMD Format field to RFH (MQRFH version 1).
-R		Set the message's MQMD Format field to RFH2 (MQRFH version 2).
-r	User_Folder	For RFH1 type message, create and set the NameValue pair. For RFH2 type message, create and set the User Folder to the text.

10.2.3.4 MQMD IniFile
The description of the MQMD IniFile can be found in Appendix A.

10.2.4 Examples

10.2.4.1 Example 1

Use PutServer to write (put) 100 binary messages to a queue from a file.

Windows
```
mqbt PutServer -p MQA1 -q TEST01.Q -n 100 -f c:\abc.pdf
```

Linux or macOS
```
mqbt PutServer -p MQA1 -q TEST01.Q -n 100 -f /abc.pdf
```

10.2.4.2 Example 2

Use PutServer to write a text string from the command-line as a 'String' message to a queue 200 times.

Windows, Linux or macOS
```
mqbt PutServer -p MQA1 -q TEST01.Q -S -n 200 -t "This is a test message."
```

10.2.4.3 Example 3

Use PutServer to write a message from a file in the Publish/Subscribe format (RFH) to a queue 500 times with a delay of 10 milliseconds between writes (puts).

Windows
```
mqbt PutServer -p MQA1 -q TEST01.Q -P -n 500 -d 10 -r "MQPSCommand RegSub
MQPSTopic" -f c:\myfile.txt
```

Linux or macOS
```
mqbt PutServer -p MQA1 -q TEST01.Q -P -n 500 -d 10 -r "MQPSCommand RegSub
MQPSTopic" -f /myfile.txt
```

10.2.4.4 Example 4

Use PutServer to write a message from a file in the JMS format (RFH2) to a queue 175 times.

Windows
```
mqbt PutServer -p MQA1 -q TEST01.Q -R -n 175 -f c:\myfile.txt
```

Linux or macOS
```
mqbt PutServer -p MQA1 -q TEST01.Q -R -n 175 -f /myfile.txt
```

10.2.4.5 Example 5

Use PutServer to write a message from a file in the JMS format (RFH2) with a user folder to a queue 400 times.

Windows
```
mqbt PutServer -p MQA1 -q TEST01.Q -R -n 400 -f c:\myfile.txt -r
"<usr><up1>ABC</up1></usr>"
```

Linux or macOS
```
mqbt PutServer -p MQA1 -q TEST01.Q -R -n 400 -f /myfile.txt -r
"<usr><up1>ABC</up1></usr>"
```

10.3 SIM Client

This section will describe how to invoke and use the SIM Client Stress Testing tool. The SIM Client will continuously send 'request' messages and wait for a reply messages. The request message's data can be from a simple input text or from a plain text file or from a Backup file (this file can contain many messages with MQMDs). A Backup file can be either a SQLite Database (*.mqsdb) or a VEQ formatted file (*.veq).

If the user does not specify a 'Reply Queue' name then SIM Client will create and use a temporary dynamic queue to receive the reply messages.

For the Stress Testing Servers (Get, Put, SIM Client & SIM Server), you can have unlimited number of them running (up to what your box can support). Also, while any of the Servers are running, the user can still use MQ Batch Toolkit.

10.3.1 Purpose

Use the **SIMClient** command to write a message from a request-queue then read a message from a reply queue. The message text for the reply message can be inputted from the command prompt (shell) or be read from a file. The message can be setup in 1 of 4 ways: binary, string, RFH (Pub/Sub) or RFH2 (JMS).

10.3.2 Syntax

```
mqbt SIMClient [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name -q
Queue_Name {-f Input_File_Name -t Text_String} -n Max_Message_Count [-o
ReplyQueueName] [-i mqmd_File_Name] [-Z] [-S] [-P] [-R] [-r User_Folder]
```

10.3.3 Parameters

10.3.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-q	Queue_Name	The name of the queue (Note: Queue names are case sensitive.)
-n	Number_of_Messages	Maximum number of messages to be read from the queue before terminating.

10.3.3.2 Requires at least 1 of the following parameters:

Key	Parameter	Description
-f	Path_File_Name	The full path and filename of the input file.
-t	Text_Message	Text string to be used as the message data (enclose text string in quotes).

10.3.3.3 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-i	mqmd_File_Name	The full path and filename of the file with the MQMD values
-o	Reply_Queue_Name	The name of the reply queue.
-d	Put_Delay	Delay between Puts in milliseconds (ms).
-Z		Perform the Get by matching Correlation ID.
-S		Set the message's MQMD Format field to string (MQSTR).
-P		Set the message's MQMD Format field to RFH (MQRFH version 1).
-R		Set the message's MQMD Format field to RFH2 (MQRFH version 2).
-r	User_Folder	For RFH1 type message, create and set the NameValue pair. For RFH2 type message, create and set the User Folder to the text.

10.3.3.4 MQMD IniFile
The description of the MQMD IniFile can be found in Appendix A.

10.3.4 Examples

10.3.4.1 Example 1

Use SIMClient to send 100 binary messages and wait for each response message.

Windows

```
mqbt SIMClient -p MQA1 -q TEST01.Q -n 100 -f c:\abc.pdf
```

Linux or macOS

```
mqbt SIMClient -p MQA1 -q TEST01.Q -n 100 -f /abc.pdf
```

10.3.4.2 Example 2

Use SIMClient to send 200 text string messages and wait for each response message.

Windows, Linux or macOS

```
mqbt SIMClient -p MQA1 -q TEST01.Q -S -n 200 -t "This is a test message."
```

10.3.4.3 Example 3

Use SIMClient to send 500 messages from a file in the Publish/Subscribe format (RFH) and wait for each response message.

Windows

```
mqbt SIMClient -p MQA1 -q TEST01.Q -P -n 500 -r "MQPSCommand RegSub MQPSTopic"
-f c:\myfile.txt
```

Linux or macOS

```
mqbt SIMClient -p MQA1 -q TEST01.Q -P -n 500 -r "MQPSCommand RegSub MQPSTopic"
-f /myfile.txt
```

10.3.4.4 Example 4

Use SIMClient to send 175 messages from a file in the JMS format (RFH2) and wait for each response message.

Windows

```
mqbt SIMClient -p MQA1 -q TEST01.Q -R -n 175 -f c:\myfile.txt
```

Linux or macOS

```
mqbt SIMClient -p MQA1 -q TEST01.Q -R -n 175 -f /myfile.txt
```

10.4 SIM Server

This section will describe how to invoke and use the SIM Server Stress Testing tool. The SIM Server will continuously consume messages and for each incoming messages, send a reply message(s). The reply message's data can be from a simple input text or from a plain text file or from a Backup file (this file can contain many messages with MQMDs). A Backup file can be either a SQLite Database (*.mqsdb) or a VEQ formatted file (*.veq).

The incoming message MUST have the Reply-To-Queue and Reply-To-QMgr fields filled in. The SIM Server will use these fields to sent a reply message to. Also, the incoming message's MsgID field of the MQMD header is copied to the outgoing message's CorrelID field of the MQMD header.

For the Stress Testing Servers (Get, Put, SIM Client & SIM Server), you can have unlimited number of them running (up to what your box can support). Also, while any of the Servers are running, the user can still use MQ Batch Toolkit.

10.4.1 Purpose

Use the **SIMServer** command to read a message from a queue then write a messages to a reply queue. The message text for the reply message can be inputted from the command prompt (shell) or be read from a file. The message can be setup in 1 of 4 ways: binary, string, RFH (Pub/Sub) or RFH2 (JMS).

10.4.2 Syntax

```
mqbt SIMServer [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name -q
Queue_Name {-f Input_File_Name -t Text_String} {-d Days -h Hours -m Minutes -n
Max_Message_Count} [-i mqmd_File_Name] [-S] [-P] [-R] [-r User_Folder]
```

10.4.3 Parameters

10.4.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-q	Queue_Name	The name of the queue (Note: Queue names are case sensitive.)

10.4.3.2 Requires at least 1 of the following parameters:

Key	Parameter	Description
-f	Path_File_Name	The full path and filename of the input file.
-t	Text_Message	Text string to be used as the message data (enclose text string in quotes).

10.4.3.3 Requires at least 1 of the following parameters:

Key	Parameter	Description
-d	Days	Consumes messages for 'x' days.
-h	Hours	Consumes messages for 'x' hours.
-m	Minutes	Consumes messages for 'x' minutes.
-n	Number_of_Messages	Maximum number of messages to be read from the queue before terminating.

10.4.3.4 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-i	mqmd_File_Name	The full path and filename of the file with the MQMD values
-S		Set the message's MQMD Format field to string (MQSTR).
-P		Set the message's MQMD Format field to RFH (MQRFH version 1).
-R		Set the message's MQMD Format field to RFH2 (MQRFH version 2).
-r	User_Folder	For RFH1 type message, create and set the NameValue pair. For RFH2 type message, create and set the User Folder to the text.

10.4.3.5 MQMD IniFile

The description of the MQMD IniFile can be found in Appendix A.

10.4.4 Examples

10.4.4.1 Example 1

Use SIMServer to reply to 100 incoming messages and each reply will be a binary message.

Windows
```
mqbt SIMServer -p MQA1 -q TEST01.Q -n 100 -f c:\abc.pdf
```

Linux or macOS
```
mqbt SIMServer -p MQA1 -q TEST01.Q -n 100 -f /abc.pdf
```

10.4.4.2 Example 2

Use SIMServer to reply to 200 incoming messages and each reply will be a text string from the command prompt as a 'String' message.

Windows, Linux or macOS
```
mqbt SIMServer -p MQA1 -q TEST01.Q -S -n 200 -t "This is a test message."
```

10.4.4.3 Example 3

Use SIMServer to reply to 500 incoming messages and each reply will be from a file in the Publish/Subscribe format (RFH).

Windows
```
mqbt SIMServer -p MQA1 -q TEST01.Q -P -n 500 -f -r "MQPSCommand RegSub
MQPSTopic" c:\myfile.txt
```

Linux or macOS
```
mqbt SIMServer -p MQA1 -q TEST01.Q -P -n 500 -r "MQPSCommand RegSub MQPSTopic"
-f /myfile.txt
```

10.4.4.4 Example 4

Use SIMServer to reply to 175 incoming messages and each reply will be from a file in the JMS format (RFH2) to a queue 175 times.

Windows
```
mqbt SIMServer -p MQA1 -q TEST01.Q -R -n 175 -f c:\myfile.txt
```

Linux or macOS
```
mqbt SIMServer -p MQA1 -q TEST01.Q -R -n 175 -f /myfile.txt
```

10.4.4.5 Example 5

Use SIMServer to reply to all incoming messages and each reply will a 'String' message from a file. SIMServer will terminate after 4 hours of execution.

Windows
```
mqbt SIMServer -p MQA1 -q TEST01.Q -h 4 -f c:\myfile.txt
```

Linux or macOS
```
mqbt SIMServer -p MQA1 -q TEST01.Q -h 4 -f /myfile.txt
```

10.5 Subscribe Server

This section will describe how to invoke and use the Subscribe Server Stress Testing tool. The Subscribe Server will continuously consume messages from a topic and show statistics of its current state.

For the Stress Testing Servers, you can have unlimited number of them running (up to what your box can support). Also, while any of the Servers are running, the user can still use MQ Batch Toolkit.

10.5.1 Purpose

Use the **SS** command to remove the messages of a particular topic of a queue manager.

10.5.2 Syntax

```
mqbt SS [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name -T Topic_Name
{-d Days -h Hours -m Minutes -n Max_Message_Count}
```

10.5.3 Parameters

10.5.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-T	Topic_Name	The name of the topic (Note: Topic names are case sensitive.)

10.5.3.2 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-d	Days	Consumes messages for 'x' days.
-h	Hours	Consumes messages for 'x' hours.
-m	Hours	Consumes messages for 'x' minutes.
-n	Max_Message_Count	Maximum number of messages to be read from the topic before terminating.

10.5.4 Examples

10.5.4.1 Example 1
Consume the messages from a topic of a queue manager for 4 hours.

Windows, Linux or macOS
```
mqbt SS -p MQA1 -T test/one -h 4
```

10.5.4.2 Example 2
Consume the messages from a topic of a queue manager for 8 days, 4 hours and 35 minutes.

Windows, Linux or macOS
```
mqbt SS -p MQA1 -T test/one -d 8 -h 4 -m 35
```

10.5.4.3 Example 3
Consume the messages from a topic of a queue manager until 5000 messages have been consumed.

Windows, Linux or macOS
```
mqbt SS -p MQA1 -T test/one -n 5000
```

10.6 Publish Server

This section will describe how to invoke and use the Publish Server Stress Testing tool. The Publish Server will publish messages to a topic. It can be used to stress test a 'Server Component'. The user can control the feed by putting a delay between the MQPUTs. The messages data can be from a simple input text or from a plain text file or from a Backup file (this file can contain many messages with MQMDs). A Backup file can be either a SQLite Database (*.mqsdb) or a VEQ formatted file (*.veq).

For the Stress Testing Servers, you can have unlimited number of them running (up to what your box can support). Also, while any of the Servers are running, the user can still use MQ Batch Toolkit.

10.6.1 Purpose

Use the **PBS** command to write (put) a messages to a particular topic of a queue manager. The message text can be inputted from the command prompt (shell) or be read from a file. The message can be setup in 1 of 4 ways: binary, string, RFH (Pub/Sub) or RFH2 (JMS).

10.6.2 Syntax

```
mqbt PBS -a Path_FileName_for_CommProfileDB -p profileName -T TopicName -n
NumberOfMessages {-f Path_FileName -t text_message} [-i mqmd_File_Name] [-S]
[-P] [-R] [-r usrFolderData] [-d PutDelayInMS]
```

10.6.3 Parameters

10.6.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.
-T	Topic_Name	The name of the topic (Note: Topic names are case sensitive.)
-n	Number_of_Messages	Maximum number of messages to be read from the topic before terminating.

10.6.3.2 Requires at least 1 of the following parameters:

Key	Parameter	Description
-f	Path_File_Name	The full path and filename of the input file.
-t	Text_Message	Text string to be used as the message data (enclose text string in quotes).

10.6.3.3 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-i	mqmd_File_Name	The full path and filename of the file with the MQMD values
-d	Put_Delay	Delay between Puts in milliseconds (ms).
-S		Set the message's MQMD Format field to string (MQSTR).
-P		Set the message's MQMD Format field to RFH (MQRFH version 1).
-R		Set the message's MQMD Format field to RFH2 (MQRFH version 2).
-r	User_Folder	For RFH1 type message, create and set the NameValue pair. For RFH2 type message, create and set the User Folder to the text.

10.6.3.4 MQMD IniFile
The description of the MQMD IniFile can be found in Appendix A.

10.6.4 Examples

10.6.4.1 Example 1

Use PBS to write (put) 100 binary messages to a topic from a file.

Windows
```
mqbt PBS -p MQA1 -T test/one -n 100 -f c:\abc.pdf
```

Linux or macOS
```
mqbt PBS -p MQA1 -T test/one -n 100 -f /abc.pdf
```

10.6.4.2 Example 2

Use PBS to write a text string from the command-line as a 'String' message to a topic 200 times.

Windows, Linux or macOS
```
mqbt PBS -p MQA1 -T test/one -S -n 200 -t "This is a test message."
```

10.6.4.3 Example 3

Use PBS to write a message from a file in the Publish/Subscribe format (RFH) to a topic 500 times with a delay of 10 milliseconds between writes (puts).

Windows
```
mqbt PBS -p MQA1 -T test/one -P -n 500 -d 10 -r "MQPSCommand RegSub MQPSTopic"
-f c:\myfile.txt
```

Linux or macOS
```
mqbt PBS -p MQA1 -T test/one -P -n 500 -d 10 -r "MQPSCommand RegSub MQPSTopic"
-f /myfile.txt
```

10.6.4.4 Example 4

Use PBS to write a message from a file in the JMS format (RFH2) to a topic 175 times.

Windows
```
mqbt PBS -p MQA1 -T test/one -R -n 175 -f c:\myfile.txt
```

Linux or macOS
```
mqbt PBS -p MQA1 -T test/one -R -n 175 -f /myfile.txt
```

10.6.4.5 Example 5

Use PBS to write a message from a file in the JMS format (RFH2) with a user folder to a topic 400 times.

Windows
```
mqbt PBS -p MQA1 -T test/one -R -n 400 -f c:\myfile.txt -r
"<usr><up1>ABC</up1></usr>"
```

Linux or macOS
```
mqbt PBS -p MQA1 -T test/one -R -n 400 -f /myfile.txt -r
"<usr><up1>ABC</up1></usr>"
```

11 Monitoring Tools

This chapter will describes how to invoke and use the Monitoring Tools included with MQ Batch Toolkit.

11.1 Channel Monitor

This section will describe how to invoke and use the Channel Monitor tool. The Channel Monitor will continuously monitor the Channel Name, Connection Name, Channel Status, Channel SubState, Messages, Last Msg Date, Last Msg Time, Bytes Sent, Bytes Received, Buffers Sent, Buffers Received, Start Channel Date, Start Channel Time, MCA Status, MCA UserID and MCA Job Name attributes of a channel.

The monitoring data will written to a CSV (Comma Separated Value) file. On subsequent invocations, the user can select to append the data to the file or write the data to a new file.

Location of the Channel Monitor CSV files:

On Windows:
```
C:\Users\{UserId}\Capitalware\MQBT\CM\{QMgrName}\
```

On Linux or macOS:
```
{home}/Capitalware/MQBT/CM/{QMgrName}/
```

11.1.1 Purpose

Use the **CM** command to start the monitoring of the channels of a queue manager.

11.1.2 Syntax

```
mqbt CM [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name {-d Days -h
Minutes -m Hours} [-k mask] [-r RefreshRate] [-A]
```

11.1.3 Parameters

11.1.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.

11.1.3.2 Requires at least 1 of the following parameters:

Key	Parameter	Description
-d	Days	Consumes messages for 'x' days.
-h	Hours	Consumes messages for 'x' hours.
-m	Minutes	Consumes messages for 'x' minutes.

11.1.3.3 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-r	Refresh_Rate	The Refresh Rate (in seconds) is the value to determine how often to retrieve the monitoring data. Default value is 60 seconds.
-k	mask	The mask can be used to limit the the number of queues to be monitored. i.e. ABC.*
-A		Append the monitoring data to CSV file (do not delete the old data)

11.1.4 Examples

11.1.4.1 Example 1

Monitor the channels of a queue manager for 4 hours.

Windows, Linux or macOS

```
mqbt CM -p MQA1 -h 4
```

11.1.4.2 Example 2

Monitor the only channels that begin with "TEST.*" of a queue manager for 8 days, 4 hours and 35 minutes.

Windows

```
mqbt CM -p MQA1 -k "TEST.*" -d 8 -h 4 -m 35
```

Linux or macOS

```
mqbt CM -p MQA1 -k "TEST.*" -d 8 -h 4 -m 35
```

Sample Output for channel TEST.CHL:

```
Time,Channel_Name,Connection_Name,Chl_Status,Chl_SubState,Messages,Last_Msg_Date,Last
_Msg_Time,Bytes_Sent,Bytes_Received,Buffers_Sent,Buffers_Received,Start_Chl_Date,Star
t_Chl_Time,MCA_Status,MCA_UserID,MCA_Job_Name

13:38:42,TEST.CHL,127.0.0.1,Running,Receiving,12,2009-09-
01,13.38.42,3452,3460,14,15,2009-09-01,13.38.42,Running,rlacroix,0000179400001728

13:38:58,TEST.CHL,127.0.0.1,Running,Receiving,16,2009-09-
01,13.38.57,8028,5692,18,19,2009-09-01,13.38.42,Running,rlacroix,0000179400001728

13:39:13,TEST.CHL,127.0.0.1,Running,In MQGet,21,2009-09-
01,13.39.13,12604,8436,22,24,2009-09-01,13.38.42,Running,rlacroix,0000179400001728

13:39:28,TEST.CHL,127.0.0.1,Running,Receiving,24,2009-09-
01,13.39.27,17180,10156,26,27,2009-09-01,13.38.42,Running,rlacroix,0000179400001728
```

11.2 Event Monitor

This section will describe how to invoke and use the Event Monitor tool. The Event Monitor will continuously read, parse, format and output event messages from the SYSTEM Event Queues.

The formatted event messages will be written to a log file. On subsequent invocations, the user can select to append the data to the file or write the data to a new file.

Location of the 3 Event Monitor log files (channel_events.log, q_mgr_events.log and perfm_events.log):

On Windows:
```
C:\Users\{UserId}\Capitalware\MQBT\EM\{QMgrName}\
```

On Linux or macOS:
```
{home}/Capitalware/MQBT/EM/{QMgrName}/
```

11.2.1 Purpose

Use the **EM** command to start the monitoring of the event queues of a queue manager.

11.2.2 Syntax

```
mqbt EM [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name {-d Days -h
Minutes -m Hours} [-A]
```

11.2.3 Parameters

11.2.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file. One than one profile can be specified. Separate each profile name with a comma.

11.2.3.2 Requires at least 1 of the following parameters:

Key	Parameter	Description
-d	Days	Consumes messages for 'x' days.
-h	Hours	Consumes messages for 'x' hours.
-m	Minutes	Consumes messages for 'x' minutes.

11.2.3.3 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-A		Append the monitoring data to CSV file (do not delete the old data)

11.2.4 Examples

11.2.4.1 Example 1

Monitor the event queues of a queue manager for 4 hours.

Windows, Linux or macOS

```
mqbt EM -p MQWT1 -h 4
```

Sample log file Output:

```
Put_Date: 2019/12/02
Put_Time: 15:18:45.710
Event_Type: MQCMD_COMMAND_EVENT
Reason: MQRC_COMMAND_PCF
MQCACF_EVENT_USER_ID: roger
MQBACF_EVENT_SECURITY_ID:
1d010105000000000005150000000a3020893142ab8234f79c12feb0300000000000000000000000000
MQIACF_EVENT_ORIGIN: MQEVO_MSG
MQCACF_EVENT_Q_MGR: MQWT2
MQBACF_EVENT_ACCOUNTING_TOKEN:
16010515000000a3020893142ab8234f79c12feb030000000000000000000000b
MQCACF_EVENT_APPL_IDENTITY:
MQIACF_EVENT_APPL_TYPE: MQAT_WINDOWS_NT
MQCACF_EVENT_APPL_NAME: MQ Explorer 9.1.3
MQCACF_EVENT_APPL_ORIGIN:
MQIACF_COMMAND: MQCMD_INQUIRE_Q_MGR
MQIACF_Q_MGR_ATTRS: MQIA_PLATFORM + MQCA_Q_MGR_NAME + MQIA_CODED_CHAR_SET_ID +
MQIA_COMMAND_LEVEL

Put_Date: 2019/12/02
Put_Time: 15:18:45.720
Event_Type: MQCMD_COMMAND_EVENT
Reason: MQRC_COMMAND_PCF
MQCACF_EVENT_USER_ID: roger
MQBACF_EVENT_SECURITY_ID:
1d010105000000000005150000000a3020893142ab8234f79c12feb0300000000000000000000000000
MQIACF_EVENT_ORIGIN: MQEVO_MSG
MQCACF_EVENT_Q_MGR: MQWT2
MQBACF_EVENT_ACCOUNTING_TOKEN:
16010515000000a3020893142ab8234f79c12feb030000000000000000000000b
MQCACF_EVENT_APPL_IDENTITY:
MQIACF_EVENT_APPL_TYPE: MQAT_WINDOWS_NT
MQCACF_EVENT_APPL_NAME: MQ Explorer 9.1.3
MQCACF_EVENT_APPL_ORIGIN:
MQIACF_COMMAND: MQCMD_INQUIRE_Q_MGR
MQIACF_Q_MGR_ATTRS: MQIACF_ALL

---

Put_Date: 2019/12/02
Put_Time: 16:58:01.200
Event_Type: MQCMD_Q_MGR_EVENT
Reason: MQRC_NOT_AUTHORIZED
MQCA_Q_MGR_NAME: MQWT2
MQIACF_REASON_QUALIFIER: MQRQ_CONN_NOT_AUTHORIZED
MQCACF_USER_IDENTIFIER: baduser
MQCACF_CSP_USER_IDENTIFIER: baduser
MQIA_APPL_TYPE: MQAT_QMGR
MQCACF_APPL_NAME: com.capitalware.mqve.MQVE
MQCACH_CHANNEL_NAME: TEST.CHL
MQCACH_CONNECTION_NAME: 10.10.10.10
```

11.2.4.2 Example 2
Monitor the event queues of 3 queue managers (MQWT1, MQWT2 and MQA1) for 7 days.

Windows, Linux or macOS
```
mqbt EM -p MQWT1,MQWT2,MQA1 -d 7
```

11.3 Queue Monitor

This section will describe how to invoke and use the Queue Monitor tool. The Queue Monitor will continuously monitor the Queue Depth, Max Depth, IPPROCS, OPPROCS, Put Inhibit and Get Inhibit attributes of a queue.

The monitoring data will written to a CSV (Comma Separated Value) file. On subsequent invocations, the user can select to append the data to the file or write the data to a new file.

Location of the Queue Monitor CSV files:

On Windows:
```
C:\Users\{UserId}\Capitalware\MQBT\QM\{QMgrName}\
```

On Linux or macOS:
```
{home}/Capitalware/MQBT/QM/{QMgrName}/
```

11.3.1 Purpose

Use the **QM** command to start the monitoring of the queues of a queue manager.

11.3.2 Syntax

```
mqbt QM [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name {-d Days -h
Minutes -m Hours} [-k mask] [-r RefreshRate] [-A] [-S]
```

11.3.3 Parameters

11.3.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.

11.3.3.2 Requires at least 1 of the following parameters:

Key	Parameter	Description
-d	Days	Consumes messages for 'x' days.
-h	Hours	Consumes messages for 'x' hours.
-m	Minutes	Consumes messages for 'x' minutes.

11.3.3.3 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-r	Refresh_Rate	The Refresh Rate (in seconds) is the value to determine how often to retrieve the monitoring data. Default value is 60 seconds.
-k	mask	The mask can be used to limit the the number of queues to be monitored. i.e. ABC.*
-A		Append the monitoring data to CSV file (do not delete the old data)
-S		Include System Queues in the list of queues to be monitored.

11.3.4 Examples

11.3.4.1 Example 1

Monitor the queues of a queue manager for 4 hours.

Windows, Linux or macOS

```
mqbt QM -p MQA1 -h 4
```

11.3.4.2 Example 2

Monitor the only queues that begin with "TEST.*" of a queue manager for 8 days, 4 hours and 35 minutes.

Windows, Linux or macOS

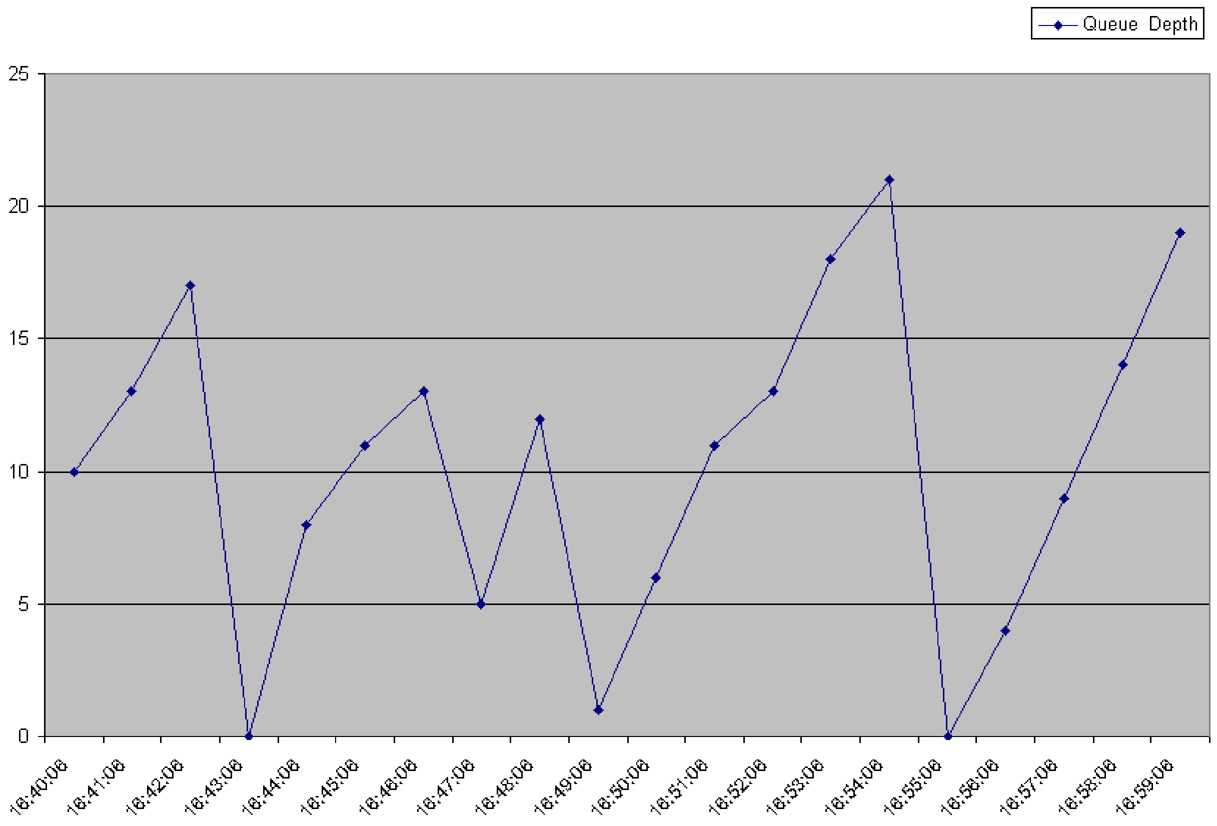
```
mqbt QM -p MQA1 -k "TEST.*" -d 8 -h 4 -m 35
```

Sample Output for queue TEST.Q1

```
Queue_Name,Queue_Depth,Max_Depth,IPPROCS,OPPROCS,Get_Inhibit,Put_Inhibit
16:40:06,10,5000,1,1,Enabled,Enabled
16:41:06,13,5000,1,1,Enabled,Enabled
16:42:06,17,5000,1,1,Enabled,Enabled
16:43:06,0,5000,1,1,Enabled,Enabled
16:44:06,8,5000,1,1,Enabled,Enabled
16:45:06,11,5000,1,1,Enabled,Enabled
16:46:06,13,5000,1,1,Enabled,Enabled
16:47:06,5,5000,1,1,Enabled,Enabled
16:48:06,12,5000,1,1,Enabled,Enabled
16:49:06,1,5000,1,1,Enabled,Enabled
16:50:06,6,5000,1,1,Enabled,Enabled
16:51:06,11,5000,1,1,Enabled,Enabled
16:52:06,13,5000,1,1,Enabled,Enabled
16:53:06,18,5000,1,1,Enabled,Enabled
16:54:06,21,5000,1,1,Enabled,Enabled
16:55:06,0,5000,1,1,Enabled,Enabled
16:56:06,4,5000,1,1,Enabled,Enabled
16:57:06,9,5000,1,1,Enabled,Enabled
16:58:06,14,5000,1,1,Enabled,Enabled
16:59:06,19,5000,1,1,Enabled,Enabled
```

Visualizing the Data

The user can use the "TEST.Q1.csv" file with a spreadsheet program and create a Line Chart. The following sample Line Chart is using just the first 2 columns of the CSV file.



11.4 Queue Statistics Monitor

This section will describe how to invoke and use the Queue Statistics Monitor tool. The Queue Statistics Monitor will continuously monitor the Put Rate (Enqueue), Get Rate (Dequeue), High Q Depth and Max Q Depth attributes of a queue.

The monitoring data will written to a CSV (Comma Separated Value) file. On subsequent invocations, the user can select to append the data to the file or write the data to a new file.

Location of the Queue Statistics Monitor CSV files:

On Windows:
```
C:\Users\{UserId}\Capitalware\MQBT\QSM\{QMgrName}\
```

On Linux or macOS:
```
{home}/Capitalware/MQBT/QSM/{QMgrName}/
```

For z/OS queue managers, you MUST enable Queue Manager Performance Events (PERFMEV) to use the Queue Stats Monitor tool against a z/OS queue manager.

11.4.1 Purpose

Use the **QSM** command to start the monitoring of the queues of a queue manager.

11.4.2 Syntax

```
mqbt QSM [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name {-d Days -h
Minutes -m Hours} [-k mask] [-r RefreshRate] [-A] [-S]
```

11.4.3 Parameters

11.4.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.

11.4.3.2 Requires at least 1 of the following parameters:

Key	Parameter	Description
-d	Days	Consumes messages for 'x' days.
-h	Hours	Consumes messages for 'x' hours.
-m	Minutes	Consumes messages for 'x' minutes.

11.4.3.3 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-r	Refresh_Rate	The Refresh Rate (in seconds) is the value to determine how often to retrieve the monitoring data. Default value is 60 seconds.
-k	mask	The mask can be used to limit the the number of queues to be monitored. i.e. ABC.*
-A		Append the monitoring data to CSV file (do not delete the old data)
-S		Include System Queues in the list of queues to be monitored.

11.4.4 Examples

11.4.4.1 Example 1

Monitor the queues of a queue manager for 4 hours.

Windows, Linux or macOS
```
mqbt QSM -p MQA1 -h 4
```

11.4.4.2 Example 2

Monitor the only queues that begin with "TEST.*" of a queue manager for 8 days, 4 hours and 35 minutes.

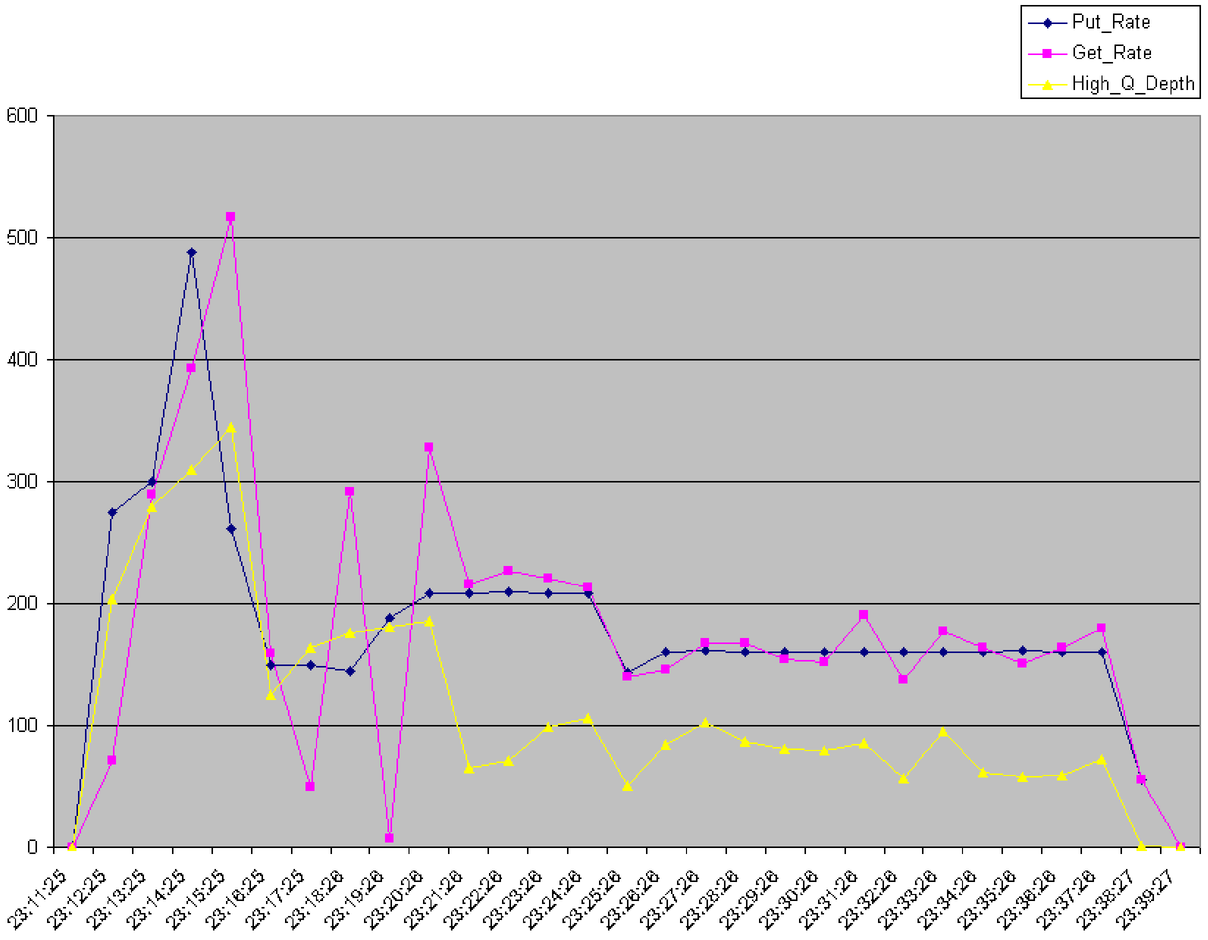
Windows, Linux or macOS
```
mqbt QSM -p MQA1 -k "TEST.*" -d 8 -h 4 -m 35
```

Sample Output for queue TEST.Q1

```
Queue_Name,Put_Rate,Get_Rate,High_Q_Depth,Max_Depth
23:11:25,0,0,0,5000
23:12:25,275,71,204,5000
23:13:25,300,289,280,5000
23:14:25,488,393,310,5000
23:15:25,262,517,344,5000
23:16:25,150,159,125,5000
23:17:25,150,49,164,5000
23:18:26,144,291,176,5000
23:19:26,188,7,181,5000
23:20:26,209,328,185,5000
23:21:26,209,216,65,5000
23:22:26,210,226,71,5000
23:23:26,209,220,99,5000
23:24:26,209,213,106,5000
23:25:26,143,140,51,5000
23:26:26,160,146,84,5000
23:27:26,161,168,102,5000
23:28:26,160,167,87,5000
23:29:26,160,154,81,5000
23:30:26,160,152,79,5000
23:31:26,160,190,86,5000
23:32:26,160,137,57,5000
23:33:26,160,177,95,5000
23:34:26,160,164,62,5000
23:35:26,161,151,58,5000
23:36:26,160,164,59,5000
23:37:26,160,179,72,5000
23:38:27,56,56,1,5000
23:39:27,0,0,0,5000
```

Visualizing the Data

The user can use the "TEST.Q1.csv" file with a spreadsheet program and create a Line Chart. The following sample Line Chart is using just the first 4 columns of the CSV file.



11.5 Queue Status Monitor

This section will describe how to invoke and use the Queue Status Monitor tool. The Queue Status Monitor will continuously monitor the queues of a queue manager displaying the Queue Name, Current Queue Depth, Open Input Count, Open Output Count, Last Put Date, Last Put Time, Last Get Date and Last Get Time.

The monitoring data will written to a CSV (Comma Separated Value) file. On subsequent invocations, the user can select to append the data to the file or write the data to a new file.

Location of the Queue Status Monitor CSV files:

On Windows:
```
C:\Users\{UserId}\Capitalware\MQBT\QSTM\{QMgrName}\
```

On Linux or macOS:
```
{home}/Capitalware/MQBT/QSTM/{QMgrName}/
```

For z/OS queue managers, you MUST enable Queue Manager Performance Events (PERFMEV) to use the Queue Stats Monitor tool against a z/OS queue manager.

11.5.1 Purpose

Use the **QSTM** command to start the monitoring of the queues of a queue manager.

11.5.2 Syntax

```
mqbt QSTM [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name {-d Days -h
Minutes -m Hours} [-k mask] [-r RefreshRate] [-A] [-S]
```

11.5.3 Parameters

11.5.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.

11.5.3.2 Requires at least 1 of the following parameters:

Key	Parameter	Description
-d	Days	Consumes messages for 'x' days.
-h	Hours	Consumes messages for 'x' hours.
-m	Minutes	Consumes messages for 'x' minutes.

11.5.3.3 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-r	Refresh_Rate	The Refresh Rate (in seconds) is the value to determine how often to retrieve the monitoring data. Default value is 60 seconds.
-k	mask	The mask can be used to limit the the number of queues to be monitored. i.e. ABC.*
-A		Append the monitoring data to CSV file (do not delete the old data)
-S		Include System Queues in the list of queues to be monitored.

11.5.4 Examples

11.5.4.1 Example 1

Monitor the queues of a queue manager for 4 hours.

Windows, Linux or macOS

```
mqbt QSTM -p MQA1 -h 4
```

11.5.4.2 Example 2

Monitor the only queues that begin with "TEST.*" of a queue manager for 8 days, 4 hours and 35 minutes.

Windows, Linux or macOS

```
mqbt QSTM -p MQA1 -k "TEST.*" -d 8 -h 4 -m 35
```

Sample Output for queue TEST.Q1

```
"Time","Queue Depth","IPPROCS","OPPROCS","Last Put Date","Last Put Time","Last
Get Date","Last Get Time"
14:01:36,7,0,0,,,,
14:02:06,7,0,0,,,,
14:02:36,7,0,0,,,,
14:03:06,7,0,0,,,,
```

11.6 Topic Monitor

This section will describe how to invoke and use the Topic Monitor tool. The Topic Monitor will continuously monitor the Time, Topic String, Topic Name, Durable, Persistence, Publications, Subscriptions, Publication Count and Subscription Count attributes of a topic.

The monitoring data will written to a CSV (Comma Separated Value) file. On subsequent invocations, the user can select to append the data to the file or write the data to a new file.

Location of the Topic Monitor CSV files:

On Windows:
```
C:\Users\{UserId}\Capitalware\MQBT\TM\{QMgrName}\
```

On Linux or macOS:
```
{home}/Capitalware/MQBT/TM/{QMgrName}/
```

11.6.1 Purpose

Use the **TM** command to start the monitoring of the topics of a queue manager.

11.6.2 Syntax

```
mqbt TM [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name {-d Days -h
Minutes -m Hours} [-k mask] [-r RefreshRate] [-A] [-S]
```

11.6.3 Parameters

11.6.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.

11.6.3.2 Requires at least 1 of the following parameters:

Key	Parameter	Description
-d	Days	Consumes messages for 'x' days.
-h	Hours	Consumes messages for 'x' hours.
-m	Minutes	Consumes messages for 'x' minutes.

11.6.3.3 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-r	Refresh_Rate	The Refresh Rate (in seconds) is the value to determine how often to retrieve the monitoring data. Default value is 60 seconds.
-k	mask	The mask can be used to limit the the number of topics to be monitored. i.e. ABC.*
-A		Append the monitoring data to CSV file (do not delete the old data)
-S		Include System Topics in the list of topics to be monitored.

11.6.4 Examples

11.6.4.1 Example 1

Monitor the topics of a queue manager for 4 hours.

Windows, Linux or macOS

```
mqbt TM -p MQA1 -h 4
```

11.6.4.2 Example 2

Monitor the only topics that begin with "test/#" of a queue manager for 8 days, 4 hours and 35 minutes.

Windows, Linux or macOS

```
mqbt TM -p MQA1 -k "test/#" -d 8 -h 4 -m 35
```

Sample Output for topic test/one:

```
"Time","Topic String","Topic
Name","Durable","Persistence","Publications","Subscriptions","Publication
Count","Subscription Count"
19:06:36,test/one,,Yes,No,Allowed,Allowed,0,0
19:07:36,test/one,,Yes,No,Allowed,Allowed,0,0
```

11.7 Subscription Monitor

This section will describe how to invoke and use the Subscription Monitor tool. The Subscription Monitor will continuously monitor the Time, Subscription Name, Topic String, Subscription User, Durable, Type, Message Count, Last Message Date and Last Message Time attributes of a subscription.

The monitoring data will written to a CSV (Comma Separated Value) file. On subsequent invocations, the user can select to append the data to the file or write the data to a new file.

Location of the Subscription Monitor CSV files:

On Windows:
```
C:\Users\{UserId}\Capitalware\MQBT\SUBM\{QMgrName}\
```

On Linux or macOS:
```
{home}/Capitalware/MQBT/SUBM/{QMgrName}/
```

11.7.1 Purpose

Use the **SUBM** command to start the monitoring of the subscriptions of a queue manager.

11.7.2 Syntax

```
mqbt SUBM [-a Path_and_FileName_for_CommProfileDB] -p Profile_Name {-d Days -h
Minutes -m Hours} [-k mask] [-r RefreshRate] [-A] [-S]
```

11.7.3 Parameters

11.7.3.1 Required Parameters

Key	Parameter	Description
-p	Profile_Name	The name of a profile contained in the CommProfileDB.properties file.

11.7.3.2 Requires at least 1 of the following parameters:

Key	Parameter	Description
-d	Days	Consumes messages for 'x' days.
-h	Hours	Consumes messages for 'x' hours.
-m	Minutes	Consumes messages for 'x' minutes.

11.7.3.3 Optional Parameters

Key	Parameter	Description
-a	Path_and_FileName	The full path and filename for a CommProfileDB.properties file.
-r	Refresh_Rate	The Refresh Rate (in seconds) is the value to determine how often to retrieve the monitoring data. Default value is 60 seconds.
-k	mask	The mask can be used to limit the the number of subscriptions to be monitored. i.e. ABC.*
-A		Append the monitoring data to CSV file (do not delete the old data)
-S		Include System Subscriptions in the list of subscriptions to be monitored.

11.7.4 Examples

11.7.4.1 Example 1

Monitor the subscriptions of a queue manager for 4 hours.

Windows, Linux or macOS

```
mqbt SUBM -p MQA1 -h 4
```

11.7.4.2 Example 2

Monitor the only subscriptions that begin with "test/#" of a queue manager for 8 days, 4 hours and 35 minutes.

Windows, Linux or macOS

```
mqbt SUBM -p MQA1 -k "test/#" -d 8 -h 4 -m 35
```

Sample Output for subscription test/one

```
"Time","Subscription Name","Topic String","Subscription
User","Durable","Type","Message Count","Last Message Date","Last Message Time"
16:20:01,MTMV_client:test/one,test/one,Guest,Yes,API,470,2017-12-20,16:09:46
16:21:01,MTMV_client:test/one,test/one,Guest,Yes,API,470,2017-12-20,16:09:46
```

12 Appendix A – MQMD IniFile

The user can use an MQMD IniFile to override the MQMD default values used when MQBT puts a message to a queue.

The MQMD IniFile has 24 keywords in the MQMD stanza.

- **AccountingToken** keyword specifies the accounting token value to be used
- **ApplicationIdData** keyword specifies the application Id data value to be used
- **ApplicationOriginData** keyword specifies the application origin data value to be used
- **CharacterSet** keyword specifies the character set value to be used
- **CorrelationId** keyword specifies the correlation Id value to be used
- **Encoding** keyword specifies the encoding value to be used
- **Expiry** keyword specifies the expiry value to be used
- **Feedback** keyword specifies the feedback value to be used
- **Format** keyword specifies the format value to be used
- **GroupId** keyword specifies the group Id value to be used
- **MessageFlags** keyword specifies the message flags value to be used
- **MessageId** keyword specifies the message Id value to be used
- **MessageSequenceNumber** keyword specifies the message sequence number value to be used
- **MessageType** keyword specifies the message type value to be used
- **Offset** keyword specifies the offset value to be used
- **Persistence** keyword specifies the persistence value to be used
  Priority keyword specifies the priority value to be used
- **PutApplicationName** keyword specifies the put application name value to be used
- **PutApplicationType** keyword specifies the put application type value to be used
- **PutDateTime** keyword specifies the put datetime value to be used
- **ReplyToQueueManagerName** keyword specifies the reply-to queue manager name value to be used
- **ReplyToQueueName** keyword specifies the reply to queue name value to be used
- **Report** keyword specifies the report value to be used
- **UserId** keyword specifies the userId value to be used

Sample mqmd_test1.ini file

```
[MQMD]
CorrelationId=ABC
UserId=tester
ApplicationOriginData=XX
```

13 Appendix B – How To Guide

13.1 Queue Manager Access Profile

Q1. How To: Add a profile to allow the MQ Batch Toolkit to connect in bindings mode to the queue manager.

A1. Command to be executed:

```
mqbt AddProfile -p MQA1 -m MQA1
```

Q2. How To: Add a profile to allow the MQ Batch Toolkit to connect in client mode to the queue manager.

A2. Command to be executed:

```
mqbt AddProfile -p MQA1 -m MQA1 -h 10.10.10.10 -n 1414 -c TEST.CHL
```

Q3. How To: Alter an exiting profile to allow the MQ Batch Toolkit to connect in bindings mode to the queue manager.

A3. Command to be executed:

```
mqbt AlterProfile -p MQA1 -q MQA1
```

Q4. How To: Alter an exiting profile to allow the MQ Batch Toolkit to connect in client mode to the queue manager.

A4. Command to be executed:

```
mqbt AlterProfile -p MQA1 -q MQA1 -h 10.10.10.10 -n 1414 -c TEST.CHL
```

Q5. How To: Delete a profile from the CommProfileDB.properties file.

A5. Command to be executed:

```
mqbt DeleteProfile -p MQA1
```

Q6. How To: List a profile from a CommProfileDB.properties file.

A6. Command to be executed:

```
mqbt ListProfile -p MQA1
```

13.2 Queue Management

Q7. How To: Backup all messages in the 'TEST01.Q' queue.

A7. Command to be executed:
On Windows:
```
mqbt Backup -p MQA1 -q TEST01.Q -f mybackup.veq
```
On Linux or macOS:
```
mqbt Backup -p MQA1 -q TEST01.Q -f mybackup.veq
```

Q8. How To: Backup 10 messages starting at message number 25 from the 'TEST01.Q' queue and delete them from the queue.

A8. Command to be executed:
On Windows:
```
mqbt Backup -p MQA1 -q TEST01.Q -f mybackup.veq -s 25 -c 25 -D
```
On Linux or macOS:
```
mqbt Backup -p MQA1 -q TEST01.Q -f mybackup.veq -s 25 -c 25 -D
```

Q9. How To: Restore all messages in the VEQ file to the 'TEST01.Q' queue.

A9. Command to be executed:
On Windows:
```
mqbt Restore -p MQA1 -q TEST01.Q -f mybackup.veq
```
On Linux or macOS:
```
mqbt Restore -p MQA1 -q TEST01.Q -f mybackup.veq
```

Q10. How To: Clear the messages in a queue of a queue manager.

A10. Command to be executed:

```
mqbt ClearQ -p MQA1 -q TEST01.Q
```

Q11. How To: Clear the messages with a particular Message ID in a queue of a queue manager.

A11. Command to be executed:

```
mqbt ClearQByID -p MQA1 -q TEST01.Q -g
414D51204D51413120202020202020209248C34020000904
```

Q12. How To: Clear the messages with a particular Correlation ID in a queue of a queue manager.

A12. Command to be executed:

```
mqbt ClearQByID -p MQA1 -q TEST01.Q -z
414243444142434400000000000000000000000000000000
```

Q13. How To: Clear the messages that both a particular Message ID and a particular Correlation ID in a queue of a queue manager.

A13. Command to be executed:

```
mqbt ClearQByID -p MQA1 -q TEST01.Q -g
414D51204D51413120202020202020209248C34020000904 -z
414243444142434400000000000000000000000000000000
```

Q14. How To: Clear the messages with a particular text string in messages of a queue of a queue manager.

A8. Command to be executed:

```
mqbt ClearQByString -p MQA1 -q TEST01.Q -t test
```

Q15. How To: Clear the messages that do not contain a particular text string starting at message number 25 of a queue of a queue manager.

A15. Command to be executed:

```
mqbt ClearQByString -p MQA1 -q TEST01.Q -t test -s 25 -X
```

Q16. How To: Clear the messages in a queue of a queue manager that are older than 4 hours.

A16. Command to be executed:

```
mqbt ClearQByTime -p MQA1 -q TEST01.Q -h 4
```

Q17. How To: Clear the messages in a queue of a queue manager that are older than 8 days, 4 hours and 35 minutes.

A17. Command to be executed:

```
mqbt ClearQByTime -p MQA1 -q TEST01.Q -d 8 -h 4 -m 35
```

Q18. How To: Clear the messages in a queue of a queue manager that are older than 72 hours and 3 minutes.

A12. Command to be executed:

```
mqbt ClearQByTime -p MQA1 -q TEST01.Q -h 72 -m 3
```

Q19. How To: Search (grep) a queue of a queue manager for a text string.

A19. Command to be executed:

```
mqbt Find -p MQA1 -t "test"
```

Q20. How To: Search (grep) a queue of a queue manager starting at message number 50 for messages without a text string.

A20. Command to be executed:

```
mqbt Find -p MQA1 -s 50 -t "test" -X
```

Q21. How To: List all queues in a queue manager.

A21. Command to be executed:

```
mqbt QList -p MQA1
```

Q22. How To: List all queues, queue types, current depth, IPPROCS and OPPROCS of a queue manager.

A22. Command to be executed:

```
mqbt QList -p MQA1 -D -T -I -O
```

Q23. How To: List only local queues that begin with 'TEST', along with queue types, current depth, IPPROCS and OPPROCS of a queue manager.

A23. Command to be executed:

```
mqbt QList -p MQA1 -m "TEST*" -D -T -I -O
```

13.3 Message Manipulation

Q24. How To: Copy all messages from the source queue to the target queue.

A24. Command to be executed:

```
mqbt Copy -p MQA1 -q TEST01.Q -t TEST02.Q
```

Q25. How To: Copy 50 messages starting at message number 20 from the source queue to the target queue.

A25. Command to be executed:

```
mqbt Copy -p MQA1 -q TEST01.Q -t TEST02.Q -s 20 -c 50
```

Q26. How To: Duplicate all messages from in the queue.

A26. Command to be executed:

```
mqbt Duplicate -p MQA1 -q TEST01.Q
```

Q27. How To: Duplicate 50 messages starting at message number 20 in the queue.

A27. Command to be executed:

```
mqbt Duplicate -p MQA1 -q TEST01.Q -s 20 -c 50
```

Q28. How To: Forward all messages from the source queue to the target queue.

A28. Command to be executed:

```
mqbt Forward -p MQA1 -q TEST01.Q -t TEST02.Q
```

Q29. How To: Forward 50 messages starting at message number 20 from the source queue to the target queue.

A29. Command to be executed:

```
mqbt Forward -p MQA1 -q TEST01.Q -t TEST02.Q -s 20 -c 50
```

Q30. How To: Insert a binary message to a queue from a file.

A30. Command to be executed:
On Windows:
```
mqbt Insert -p MQA1 -q TEST01.Q -f c:\abc.pdf
```
On Linux or macOS:
```
mqbt Insert -p MQA1 -q TEST01.Q -f /abc.pdf
```

Q31. How To: Insert a text string from the command prompt as a 'String' message to a queue.

A31. Command to be executed:

```
mqbt Insert -p MQA1 -q TEST01.Q -S -t "This is a test message."
```

Q32. How To: Insert a message from a file in the Publish/Subscribe format (RFH) to a queue.

A32. Command to be executed:
On Windows:
```
mqbt Insert -p MQA1 -q TEST01.Q -P -r "MQPSCommand RegSub MQPSTopic" -f c:\
myfile.txt
```
On Linux or macOS:
```
mqbt Insert -p MQA1 -q TEST01.Q -P -r "MQPSCommand RegSub MQPSTopic" -f
/myfile.txt
```

Q33. How To: Insert a message from a file in the JMS format (RFH2) to a queue.

A33. Command to be executed:
On Windows:
```
mqbt Insert -p MQA1 -q TEST01.Q -R -f c:\myfile.txt
```
On Linux or macOS:
```
mqbt Insert -p MQA1 -q TEST01.Q -R -f /myfile.txt
```

Q34. How To: Insert a message from a file in the JMS format (RFH2) with a user folder to a queue.

A34. Command to be executed:
On Windows:
```
mqbt Insert -p MQA1 -q TEST01.Q -R -f c:\myfile.txt -r
"<usr><up1>ABC</up1></usr>"
```
On Linux or macOS:
```
mqbt Insert -p MQA1 -q TEST01.Q -R -f /myfile.txt -r
"<usr><up1>ABC</up1></usr>"
```

Q35. How To: Delete all messages from in the queue.

A35. Command to be executed:

```
mqbt Delete -p MQA1 -q TEST01.Q
```

Q36. How To: Delete 50 messages starting at message number 20 in the queue.

A36. Command to be executed:

```
mqbt Delete -p MQA1 -q TEST01.Q -s 20 -c 50
```

Q37. How To: Import a binary message to a queue from a file.

A37. Command to be executed:
On Windows:
```
mqbt Import -p MQA1 -q TEST01.Q -f c:\abc.pdf
```
On Linux or macOS:
```
mqbt Import -p MQA1 -q TEST01.Q -f /abc.pdf
```

Q38. How To: Import a 'string' message to a queue from a file.

A38. Command to be executed:
On Windows:
```
mqbt Import -p MQA1 -q TEST01.Q -S -f textfile.txt
```
On Linux or macOS:
```
mqbt Import -p MQA1 -q TEST01.Q -S -f textfile.txt
```

Q39. How To: Import a message from a file in the Publish/Subscribe format (RFH) to a queue.

A39. Command to be executed:
On Windows:
```
mqbt Import -p MQA1 -q TEST01.Q -P -r "MQPSCommand RegSub MQPSTopic" -f c:\myfile.txt
```
On Linux or macOS:
```
mqbt Import -p MQA1 -q TEST01.Q -P -r "MQPSCommand RegSub MQPSTopic" -f /myfile.txt
```

Q40. How To: Import a message from a file in the JMS format (RFH2) to a queue.

A40. Command to be executed:
On Windows:
```
mqbt Import -p MQA1 -q TEST01.Q -R -f c:\myfile.txt
```
On Linux or macOS:
```
mqbt Import -p MQA1 -q TEST01.Q -R -f /myfile.txt
```

Q41. How To: Import a message from a file in the JMS format (RFH2) with a user folder to a queue.

A41. Command to be executed:
On Windows:
```
mqbt Import -p MQA1 -q TEST01.Q -R -f c:\myfile.txt -r "<usr><up1>ABC</up1></usr>"
```
On Linux or macOS:
```
mqbt Import -p MQA1 -q TEST01.Q -R -f /myfile.txt -r "<usr><up1>ABC</up1></usr>"
```

Q42. How To: Export all messages in the 'TEST01.Q' queue.

A42. Command to be executed:
On Windows:
```
mqbt Export -p MQA1 -q TEST01.Q -f textfile.txt
```
On Linux or macOS:
```
mqbt Export -p MQA1 -q TEST01.Q -f textfile.txt
```

Q43. How To: Export 10 messages starting at message number 25 from the 'TEST01.Q' queue and delete them from the queue.

A43. Command to be executed:
On Windows:
```
mqbt Export -p MQA1 -q TEST01.Q -f textfile.txt -s 25 -c 25 -D
```
On Linux or macOS:
```
mqbt Export -p MQA1 -q TEST01.Q -f textfile.txt -s 25 -c 25 -D
```

Q44. How To: Export all messages doing a destructive get (deleting the messages) with the option of 'Convert on Get' from the 'TEST01.Q' queue.

A44. Command to be executed:
On Windows:
```
mqbt Export -p MQA1 -q TEST01.Q -f textfile.txt -D -C
```
On Linux or macOS:
```
mqbt Export -p MQA1 -q TEST01.Q -f textfile.txt -D -C
```

Q45. How To: Read all messages from the queue.

A45. Command to be executed:

```
mqbt Read -p MQA1 -q TEST01.Q
```

Q46. How To: Read all messages in Hex format from the queue.

A46. Command to be executed:

```
mqbt Read -p MQA1 -q TEST01.Q -H
```

Q47. How To: Read all messages in EBCDIC Hex format from the queue.

A47. Command to be executed:

```
mqbt Read -p MQA1 -q TEST01.Q -E
```

Q48. How To: Read all messages from the queue and display the message's MQMD (message descriptor).

A48. Command to be executed:

```
mqbt Read -p MQA1 -q TEST01.Q -M
```

13.4 Stress Testing Tools

Q49. How To: Consume the messages in a queue of a queue manager for 4 hours.

A49. Command to be executed:

```
mqbt GetServer -p MQA1 -q TEST01.Q -h 4
```

Q50. How To: Consume the messages in a queue of a queue manager for 8 days, 4 hours and 35 minutes.

A50. Command to be executed:

```
mqbt GetServer -p MQA1 -q TEST01.Q -d 8 -h 4 -m 35
```

Q51. How To: Consume the messages in a queue of a queue manager until 5000 messages have been consumed.

A51. Command to be executed:

```
mqbt GetServer -p MQA1 -q TEST01.Q -n 5000
```

Q52. How To: Use PutServer to write (put) 100 binary messages to a queue from a file.

A52. Command to be executed:
On Windows:
```
mqbt PutServer -p MQA1 -q TEST01.Q -n 100 -f c:\abc.pdf
```
On Linux or macOS:
```
mqbt PutServer -p MQA1 -q TEST01.Q -n 100 -f /abc.pdf
```

Q53. How To: Use PutServer to write a text string from the command prompt as a 'String' message to a queue 200 times.

A53. Command to be executed:

```
mqbt PutServer -p MQA1 -q TEST01.Q -S -n 200 -t "This is a test message."
```

Q54. How To: Use PutServer to write a message from a file in the Publish/Subscribe format (RFH) to a queue 500 times with a delay of 10 milliseconds between writes (puts).

A54. Command to be executed:
On Windows:
```
mqbt PutServer -p MQA1 -q TEST01.Q -P -n 500 -d 10 -f c:\myfile.txt
```
On Linux or macOS:
```
mqbt PutServer -p MQA1 -q TEST01.Q -P -n 500 -d 10 -f /myfile.txt
```

Q55. How To: Use PutServer to write a message from a file in the JMS format (RFH2) to a queue 175 times.

A55. Command to be executed:
On Windows:
```
mqbt PutServer -p MQA1 -q TEST01.Q -R -n 175 -f c:\myfile.txt
```
On Linux or macOS:
```
mqbt PutServer -p MQA1 -q TEST01.Q -R -n 175 -f /myfile.txt
```

Q56. How To: Use PutServer to write a message from a file in the JMS format (RFH2) with a user folder to a queue 400 times.

A56. Command to be executed:
On Windows:
```
mqbt PutServer -p MQA1 -q TEST01.Q -R -n 400 -f c:\myfile.txt -r
"<usr><up1>ABC</up1></usr>"
```
On Linux or macOS:
```
mqbt PutServer -p MQA1 -q TEST01.Q -R -n 400 -f /myfile.txt -r
"<usr><up1>ABC</up1></usr>"
```

Q57. How To: Use SIMServer to reply to 100 incoming messages and each reply will be a binary message.

A57. Command to be executed:
On Windows:
```
mqbt SIMServer -p MQA1 -q TEST01.Q -n 100 -f c:\abc.pdf
```
On Linux or macOS:
```
mqbt SIMServer -p MQA1 -q TEST01.Q -n 100 -f /abc.pdf
```

Q58. How To: Use SIMServer to reply to 200 incoming messages and each reply will be a text string from the command prompt as a 'String' message.

A58. Command to be executed:

```
mqbt SIMServer -p MQA1 -q TEST01.Q -S -n 200 -t "This is a test message."
```

Q59. How To: Use SIMServer to reply to 500 incoming messages and each reply will be from a file in the Publish/Subscribe format (RFH).

A59. Command to be executed:
On Windows:
```
mqbt SIMServer -p MQA1 -q TEST01.Q -P -n 500 -f c:\myfile.txt
```
On Linux or macOS:
```
mqbt SIMServer -p MQA1 -q TEST01.Q -P -n 500 -f /myfile.txt
```

Q60. How To: Use SIMServer to reply to 175 incoming messages and each reply will be from a file in the JMS format (RFH2) to a queue 175 times.

A35. Command to be executed:
On Windows:
```
mqbt SIMServer -p MQA1 -q TEST01.Q -R -n 175 -f c:\myfile.txt
```
On Linux or macOS:
```
mqbt SIMServer -p MQA1 -q TEST01.Q -R -n 175 -f /myfile.txt
```

Q61. How To: Use SIMServer to reply to all incoming messages and each reply will a 'String' message from a file. SIMServer will terminate after 4 hours of execution.

A61. Command to be executed:
On Windows:
```
mqbt SIMServer -p MQA1 -q TEST01.Q -h 4 -f c:\myfile.txt
```
On Linux or macOS:
```
mqbt SIMServer -p MQA1 -q TEST01.Q -h 4 -f /myfile.txt
```

13.5 Monitoring Tools

Q62. How to use the Queue Monitor to monitor the queues of a queue manager for 4 hours?

A62. Command to be executed:

```
mqbt QM -p MQA1 -h 4
```

Q63. How to use the Queue Stats Monitor to monitor the queues of a queue manager for 4 hours?

A63. Command to be executed:

```
mqbt QSM -p MQA1 -h 4
```

14 Appendix C – Frequently Asked Questions (FAQ)

Q. Is any of the output logged?

A. Yes, the output is generated with Log4J and by default, it is written to the mqbt.log file. It is located at:
i.e.
On Windows:
`C:\Capitalware\MQBT\log\mqbt.log`
On Linux or macOS:
`Capitalware/MQBT/log/mqbt.log`

The configuration of Log4J has already been done and is in the log4j.properties file. It is located at:
i.e.
On Windows:
`C:\Capitalware\MQBT\log4j.properties`
On Linux or macOS:
`Capitalware/MQBT/log4j.properties`

Q. How do I get rid of the logging messages on the screen / console?

A. If the user does not want the logging information to be displayed on the screen / console then update log4j.properties file and modify the log4j.rootCategory parameter to be as follows:
i.e.
On Windows:
`C:\Capitalware\MQBT\log4j.properties`
On Linux or macOS:
`Capitalware/MQBT/log4j.properties`

FROM:
```
log4j.category.com.capitalware.mqbt.MQBT=INFO, mqbt, stdout
```
TO:
```
log4j.category.com.capitalware.mqbt.MQBT=INFO, mqbt
```

Q. Can I use the 'Queue Manager Access Profile' file (CommProfileDB.properties) from MQ Batch Toolkit or MQ Batch Toolkit with MQ Batch Toolkit?

A. Yes, if you are a licensed user of both MQ Batch Toolkit (MQ Batch Toolkit) and MQ Batch Toolkit then you will be able use an existing CommProfileDB.properties file with MQ Batch Toolkit.

Q. Can I use the '*.VEQ' files created from MQ Batch Toolkit or MQ Batch Toolkit with MQ Batch Toolkit?

A. Yes, the Backup / Restore VEQ file format is standardized across products. Hence, you will be able to create a VEQ file with one tool and use it with another Capitalware product.

Q. I purchased a license and received the license key, what do I do with it?

A. Follow the following instructions to input your license to MQ Batch Toolkit.
Edit the MQBT.ini file located in the products home directory.
i.e.
On Windows:
```
C:\Capitalware\MQBT\MQBT.ini
```
On Linux or macOS:
```
Capitalware/MQBT/MQBT.ini
```

Your license will look something like: 0000AAAA0000BBBB (Note: This is a sample license only and will NOT work). In the [Defaults] section of the **MQBT.ini** file, insert your license key as follows (remove the dashes from the license key):

```
[Defaults]
License=0000AAAA0000BBBB
Name=John Doe
Email=john.doe@acme.com
```

Save the file and then run MQ Batch Toolkit's ***Register*** function. MQ Batch Toolkit will retrieve the Access Code from Capitalware's web server and then the user will be able to define and use an unlimited number of queue managers.

Q. What does reason code 2058 mean?

A. Generally, 2058 means that you have an invalid queue manager name or that the host does not have that particular queue manager. Check the spelling of the queue manager name that you inputted. Also, queue manager names are case sensitive (e.g. MQA1 is not the same as mqa1).

Q. What does reason code 2059 mean?

A. Generally, 2059 means that the queue manager is down. Try restarting the queue manager and command server on the remote box. Reason code 2059 can also occur if you inputted an incorrect port number. Therefore, verify that the port number is correct.

Q. What does reason code 2009 mean?

A. Generally, 2009 means that the channel name is incorrect. Verify that you have inputted the correct server connection channel name. Also, channel names are case sensitive (e.g. JAVA.CHL is not the same as java.chl).

Q. What does reason code 2035 mean?

A. The reason code 2035 means that the user's UserId is not authorized to access the queue on the remote (or local) queue maanger. Ask your IBM MQ Administrator to grant your UserId the necessary authority to access the queue.

Q. Can I use MQ Batch Toolkit with my mainframe (OS/390 or z/OS) queue managers?

A. Yes. But you need to have the CAF (Client Attachment Feature) installed on z/OS (mainframe). On the mainframe, go to SDSF and look in the xxxxCHIN (where xxxx is the queue manager name) output for a message:

```
CSQX099I - xxxx CSQXGIP Client attachment feature available
```

If this message doesn't show up, then you don't have the feature installed. The CAF is a required for any client program (including MQ Batch Toolkit) to connect directly to a mainframe queue manager.

Q. Can I install MQ Batch Toolkit on a network drive (LAN) and run it from that location?

A. Yes. MQ Batch Toolkit will happily run from a network drive. Each user's preferences will be stored in their own "Home Directory".

On Windows, the home directory is the **%USERPROFILE%** folder:
i.e.
`C:\Users\{UserID}\`

On Linux and macOS, the home directory is the user's standard logon directory
i.e.
`/export/home/{UserID}/`

Q. Can I share my queue manager connection information for MQ Batch Toolkit with another user or put it on a network drive (LAN)?

A. Yes. MQ Batch Toolkit stores the queue manager connection information in a file called: CommProfileDB.properties.

15 Appendix D – MQ Batch Toolkit Upgrade Procedures

To upgrade an existing installation of MQ Batch Toolkit, please do the following in the appropriate section below.

15.1 Windows Upgrade

- Stop all instances of MQBT
- Backup all MQBT data files in the MQBT install directory
- Delete the MQBT install directory
- Run the MQ Batch Toolkit installer (mqbt-setup-withjre.exe)
- Restore the MQBT data files if necessary

15.2 Linux Upgrade

- Stop all instances of MQBT
- Backup all MQBT data files in the MQBT install directory
- Delete the MQBT install directory
- Run the MQ Batch Toolkit installer (mqbt-setup-linux.bin)
- Restore the MQBT data files if necessary

15.3 macOS Upgrade

- Stop all instances of MQBT
- Backup all MQBT data files in the MQBT install directory
- Delete the MQBT install directory
- Unzip and un-tar the mqbt-macOS.tar.zip file
- Restore the MQBT data files if necessary

16 Appendix E – Support

The support for MQ Batch Toolkit can be found at the following location:

Online Help Desk Ticketing System at
www.capitalware.com/phpst/

By email at:
support@capitalware.com

By regular mail at:

Capitalware Inc.
Attn: MQ Batch Toolkit Support
Unit 11, 1673 Richmond Street, PMB524
London, Ontario Canada
N6G 2N3

17 Appendix F – Summary of Changes

- MQ Batch Toolkit v3.2.2
  - Fixed an issue with handling PCF MQCFT_XR_SUMMARY messages from z/OS queue managers.
  - Added code to output the directory used to write the CSV files from the MQ Monitoring Tools.

- MQ Batch Toolkit v3.2.1
  - Added code to better handle missing channel exit and/or incorrect path to channel exit.
  - Fixed a bug in the registration process
  - Fixed an issue with Generate Report not properly creating the character display portion of the HEX output in the PDF.
  - Fixed an issue when a connection fails but MQBT attempts to check if the Command Server is running.

- MQ Batch Toolkit v3.2.0
  - Updated the QList and TopicList function to have '-S' parameter that will allow SYSTEM queues or topics to be included in the output list
  - Fixed an issue with parsing event (PCF) messages and displaying them.
  - Updated SSL/TLS support
  - Enhanced the error message regarding an expired license key
  - Added warning message that the trial-only release cannot be registered.

- MQ Batch Toolkit v3.1.0
  - Updated Backup and BackupTopic functions to support writing the messages to a SQLite database.
  - Updated Restore and RestoreTopic functions to support reading the messages from a SQLite database.
  - Updated PutServer, SIMClient, SIMServer and PublishServer functions to support reading the messages from an SQLite database.
  - Changed the web call to capitalware.com server for license registration from http to https.
  - Updated AddProfile and AlterProfile to support UserId compatibility mode to send the UserId and Password as done in releases prior to IBM MQ V8.

- MQ Batch Toolkit v3.0.0
  - Added BackupTopic function to subscribe to a topic and write the messages to a VEQ file.
  - Added RestoreTopic function to read a VEQ file and publish the messagse to a topic.
  - Added ReportTopic function to generate a report (PDF, RTF or HTML) from topic messages.
  - Added Subscribe function to subscribe to a topic and output the message data to the screen (console).
  - Added Publish function to publish a message to a topic.
  - Added TopicList function that will retrieve a list of topics from a local or remote queue manager.

  - Added SubscribeServer (SS) function that will continuously consume messages of a topic.
  - Added PublishServer (PBS) function that will publish (put) messages to a topic.
  - Added TopicMonitor (TM) function that will continuously monitor the topics of a queue manager writing the information to a CSV (Comma Separated Value) file.
  - Added SubscriptionMonitor (SUBM) function that continuously monitor the subscriptions of a queue manager writing the information to a CSV (Comma Separated Value) file.
  - Added QueueStatusMonitor (QSTM) function that will continuously monitor the queues of a queue manager writing the information to a CSV (Comma Separated Value) file.
  - Added Register function to allow an end-user to register the license key and retrieve an Access Code.
  - Added '-C' parameter (Convert on Get) for ClearQByString function.
  - Added '-N' parameter (Named Properties) for Report function.
  - Added JVM environment variables: com.ibm.mq.cfg.useIBMCipherMappings=false & com.ibm.jsse2.disableSSLv3=false
  - Added more SSL/TLS CipherSpecs
  - Added Send and receive exits for queue manager access profile (AddProfile & AlterProfile).
  - Updated GetServer to output the message payload to the console of the message received.
  - Fixed the error message for path to security exit JAR file for AddProfile and AlterProfile
  - MQ Batch Toolkit is now built as a 64-bit executable for Linux, macOS and Windows.
  - Updated docs (English only)

- MQ Batch Toolkit v2.0.4
  - Altered code for the Insert command to handle extremely large files (i.e. 900MB) for '1 message per line' of the file feature
  - Altered code for QM, QSM & CM commands so that they do not do an extra wait before exiting.
  - For GetEmail, Import, Insert, PutServer & SIMClient commands, changed RFH2 message type from "jms_bytes" to "jms_text"
  - Fixed a bug in QM, QSM & CM commands when displaying the refreshRate.
  - Updated docs (English only)

- MQ Batch Toolkit v2.0.3
  - On Windows, switched JVM to Excelsior Jet v1.6.0_41
  - Updated docs (English only)

- MQ Batch Toolkit v2.0.2
  - Fixed a recursive bug during connection to a queue manager
  - Updated docs (English only)

- MQ Batch Toolkit v2.0.1
  - Updated MQBT to properly handle MQRFH2 messages under WMQ v7.*
  - Updated docs (English only)

- MQ Batch Toolkit v2.0.0
    - Major rewrite of the command-line option processing.
    - Major rewrite of the internal message processing engine.
    - Added SendEmail function to get one or more messages from a queue and send each MQ message as an email message (via SMTP).
    - Added GetEmail function to retrieve one or more email messages (via POP) and put each email message as an MQ message to an MQ queue.
    - Added Report function to generate a formatted document containing one of more messages. The formatted report can be a PDF, RTF or HTML file.
    - Added '-i' parameter to the Insert, Import, PutServer, SIMClient and SIMServer functions that specifies a file with the MQMD values.
    - Added "-X" parameter to the Copy and Forward functions to remove any MQ headers during the export process.
    - Updated the MQBT and MQBT64 shell scripts to support IBM i (OS/400)
    - Fully tested and supported for IBM MQ v7.1
    - Updated docs (English only)

- MQ Batch Toolkit v1.3.2
    - Added Password field for AddProfile and AlterProfile functions
    - Added WMQ v7 event queues for the Event Monitor (EM) function
    - Fixed an issue with Forward function when '-s' parameter is greater than 1
    - Fixed a null pointer issue with AlterProfile function
    - Fixed a bug in CheckUp function when processing 'Process' definitions.
    - Fully tested and supported for Windows 7 Professional

- MQ Batch Toolkit v1.3.1
    - Added “-X” parameter to the Export function to remove any MQ headers during the export process.
    - Fixed a CipherSpec issue for AddProfile and AlterProfile
    - Fixed a bug with CheckUp and Cluster Queues

- MQ Batch Toolkit v1.3.0
    - Added new feature called Checkup to validate / verify attributes of MQ objects
    - Added new feature called Channel Monitor and it logs the monitor data to a CSV (Comma Separated Value) file
    - Added new feature called Event Monitor and it logs the event messages to a log file
    - Added new feature called PortScan
    - Added reconnection logic for Queue Monitor, Queue Stats Monitor, Channel Monitor and Event Monitor
    - Fixed a bug in the ClearQ function
    - Fixed a Java heap issue with Restore function
    - Updated and converted documentation to PDF format (English only)

- MQ Batch Toolkit v1.2.3
    - Added SyncPoint control to the Export function
    - Updated docs (English only)

- MQ Batch Toolkit v1.2.2
  - Added new feature called Queue Monitor and it logs the monitor data to a CSV (Comma Separated Value) file
  - Added new feature called Queue Stats Monitor and it logs the monitor data to a CSV (Comma Separated Value) file
  - Added a 'Trailer Text' parameter to the Export function
  - Updated docs (English only)

- MQ Batch Toolkit v1.2.1
  - Added new feature called: SIM Client
  - Fixed error messages for invalid command-line parameters
  - Fixed null values in CommProfileDB.properties
  - Updated docs (English only)

- MQ Batch Toolkit v1.2.0
  - Added SSL support.
  - Fixed RFH and RFH2 header length issue.
  - Added support for user folder for RFH messages.
  - Fixed a bug in Clear Queue and List Queue functions to correctly handle UserID and Security Exit.
  - Fully tested and supported for Java v6
  - Fully tested and supported for Windows Vista Business
  - Updated docs (English only)

- MQ Batch Toolkit v1.1.8
  - Added support for MQHSAP & SMQBAD message types (SAP) for the Read function.
  - Fixed a bug in AddProfile and AlterProfile functions to correctly handle UserID and Security Exit.
  - Updated docs (English only)

- MQ Batch Toolkit v1.1.7
  - Fixed a bug in the exception handling code for Backup and Export functions.
  - Fixed a bug in the mqbt.sh shell script.
  - Updated docs (English only)

- MQ Batch Toolkit v1.1.6
  - Added code to verify a VEQ file for Restore, PutServer and SIMServer.
  - Added code to verify a VEQ file and its file length for Backup with the append option.
  - Added code to make sure the VEQ header is updated after each message is written to the file.
  - Fixed a bug in PutServer when using 'max message to put' for VEQ file.
  - Updated docs (English only)

- MQ Batch Toolkit v1.1.5
  - Added SyncPoint (Unit of Work) for both Backup and Restore
  - Fixed a bug in Bacup and Restore to make sure the path and / or filename are valid.

  - New full language support for Romanian (ro).
  - Updated docs (English only)

- MQ Batch Toolkit v1.1.1
  - Fixed a bug with the 'stripping of the DLH' from a message.
  - Updated docs (English only)

- MQ Batch Toolkit v1.1.0
  - Added support for 3rd party Security Exit
  - Moved the UserID field to the Queue Manager Access Profile.
  - Updated docs (English only)

- MQ Batch Toolkit v1.0.0
  - Initial release.

18 Appendix G1 – License Agreement - Unregistered

LIMITED WARRANTY

THE PROGRAM IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL THE AUTHOR OR AUTHORS BE LIABLE TO YOU FOR ANY DAMAGES, INCLUDING INCIDENTAL OR CONSEQUENTIAL DAMAGES, ARISING OUT OF THE USE OF THE PROGRAM, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

YOU ACKNOWLEDGE THAT YOU HAVE READ THIS LICENSE, UNDERSTAND IT AND AGREE TO BE BOUND BY ITS TERMS AS THE COMPLETE AND EXCLUSIVE STATEMENT OF THE AGREEMENT BETWEEN US, SUPERSEDING ANY PROPOSAL OR PRIOR AGREEMENT, ORAL OR WRITTEN, AND ANY OTHER COMMUNICATIONS BETWEEN US RELATING TO THE SUBJECT MATTER OF THIS LICENSE.

MQ Batch Toolkit is a shareware program and is provided at no charge to the user for evaluation. The purpose of shareware software is to provide personal computer users with quality software on a "try before you buy" basis, however payment is still required for continued use of the product.

If you find this program useful and continue to use it after a the trial period, you must make a registration payment (see the registration instructions for details). This registration fee will license one user to use one copy of MQ Batch Toolkit on any one computer at any one time. All users will receive a copy of the latest release when they register, or it will be made available for downloading, and free technical support.

Commercial users must register and pay for their copies within 30 days of first use or their license is withdrawn.

Anyone distributing this product for any kind of remuneration must first contact Capitalware Inc. for authorization.

You may distribute this software to friends and colleagues but you must include all files in the original distribution. Please encourage them to register their copy if they find that they make use of it.

19 Appendix G2 – License Agreement - Registered

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 user, or group of users for Enterprise License. This Agreement requires that each user of the SOFTWARE be Licensed, either individually, or as part of a group. An Enterprise License provides for an unlimited number of users to use this SOFTWARE at any time at the specific company. The individual license does not provide for concurrent user Licensing. Each user of this SOFTWARE must be covered either individually, or as part of a group 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 users only. The license, whether individual or enterprise, is for a 1-year subscription which includes the latest release of the SOFTWARE and support.

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 MQBT.REG, MQBT.PROPERTIES, MQBT.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.

20 Appendix H – Notices

<u>Trademarks:</u>