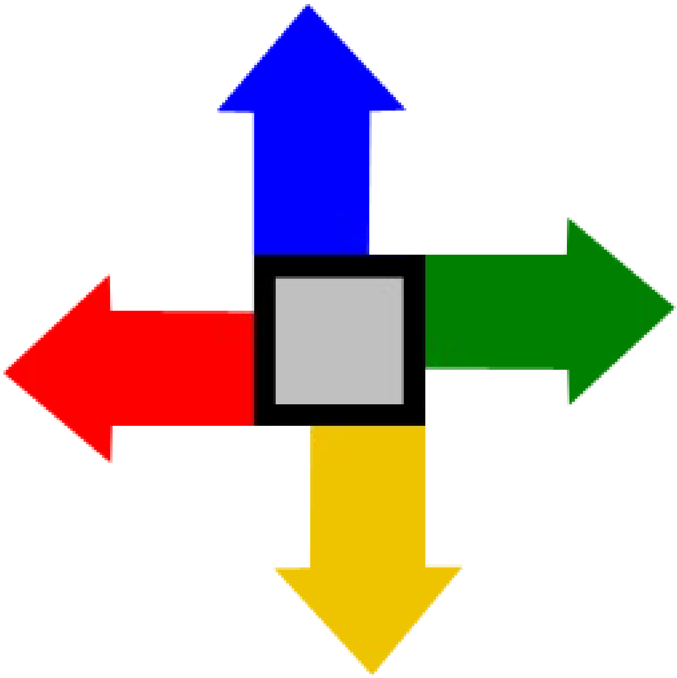
MQ File Mover Installation and Operation Manual



Capitalware Inc.
1673 Richmond Street, Suite 524
London, Ontario, Canada
N6G 2N3
sales@capitalware.biz
http://www.capitalware.biz

Table of Contents

1 INTRODUCTION........1

1.1 Overview........1
1.2 Executive Summary........3
1.3 Context Diagram (Logical View)........4
1.4 Prerequisites........5
  1.4.1 Java........5
  1.4.2 WebSphere MQ........5
1.5 File Definitions........5

2 INSTALLING MQ FILE MOVER........6

2.1 Windows Installation........6
  2.1.1 Installing MQFM as a Windows Service........6
  2.1.2 Starting or Stopping the MQFM Service........7
  2.1.3 Uninstalling the MQFM Service........7
2.2 Unix and Linux Installation........7
2.3 IBM i Installation........8

3 STARTING MQFM........9

3.1 Command-line parameters........9
3.2 Triggered Execution........9
3.3 Manual Execution........9
3.4 Scheduled Execution........9
3.5 Windows Service........9
3.6 Sample MQFM Workflow XML Files........10
  3.6.1 Sample #1........10
  3.6.2 Sample #2........11
  3.6.3 Sample #3........12
  3.6.4 Sample #4........13
  3.6.5 Sample #5........14

4 MQFM_WORKFLOW XML FILE........15

4.1 <MQFM_Workflow> Root Element........15
4.2 <Global> Element........15
  4.2.1 Attributes........15
  4.2.2 <Property> Element........15
  4.2.3 <LogFile> Element........17
  4.2.4 <OnError> Element........17
  4.2.5 <MQ> Element........17
4.3 <Actions> Elements........18
  4.3.1 <Send> Element........18
  4.3.2 <Watch> Element........21
  4.3.3 <Receive> Element........24
  4.3.4 <PutQuit> Element........26
  4.3.5 <Execute> Element........27

4.3.6 <If> Element......28
4.3.7 <Launch> Element......35
4.3.8 <Schedule> Element......36
4.3.9 <SendEmail> Element......37
4.3.10 <Sleep> Element......38
4.3.11 <Append> Element......39
4.3.12 <Copy> Element......40
4.3.13 <DecryptFile> Element......41
4.3.14 <Delete> Element......42
4.3.15 <EncryptFile> Element......43
4.3.16 <Merge> Element......44
4.3.17 <MergeSort> Element......45
4.3.18 <Move> Element......46
4.3.19 <Rename> Element......47
4.3.20 <ReplaceText> Element......48
4.3.21 <Sort> Element......49
4.3.22 <Tar> Element......50
4.3.23 <Touch> Element......51
4.3.24 <UnTar> Element......52
4.3.25 <UnZip> Element......53
4.3.26 <Zip> Element......54

5 MQFM_EMAIL XML FILE......55

5.1 <MQFM_EMAIL> ROOT ELEMENT......55
5.2 <SMTPHOST> ELEMENT......55
5.3 <SMTPPORT> ELEMENT......55
5.4 <SMTPUSERID> ELEMENT......55
5.5 <SMTPPASSWORD> ELEMENT......55
5.6 <FROMEMAILADDRESS> ELEMENT......55
5.7 <TOEMAILADDRESS> ELEMENT......55
5.8 <SUBJECT> ELEMENT......56
5.9 <MESSAGETEXT> ELEMENT......56
5.10 SAMPLE......56

6 MQFM_JOB XML FILE......57

6.1 <MQFM_JOB> ROOT ELEMENT......57
6.2 <JOB> ELEMENT......57
6.3 SAMPLE......58

7 MQFM_MQ XML FILE......59

7.1 <MQFM_MQ> ROOT ELEMENT......59
7.2 SAMPLE......60

8 MQFM_SCHEDULE XML FILE......61

8.1 <MQFM_SCHEDULE> ROOT ELEMENT......61
8.2 <SCHEDULE> ELEMENT......61
8.2.1 Attributes......61
8.2.2 Elements......61

8.3 SAMPLE....62

9 MQFM_WATCH XML FILE....63

9.1 <MQFM_WATCH> ROOT ELEMENT....63
9.2 <POLLINTERVAL> ELEMENT....63
9.3 <WATCH> ELEMENT....63
  9.3.1 Attributes....63
  9.3.2 Elements....64
9.4 SAMPLE....66

10 MQFM TOKENS....67

10.1 TOKENS....67
  10.1.1 Date / Time Tokens....67
  10.1.2 File Tokens....68
  10.1.3 Index Token....68
  10.1.4 General Tokens....68
10.2 EXAMPLES....69

11 APPENDIX A – CUSTOM LOGGING....70

12 APPENDIX B – MQ FILE MOVER UPGRADE PROCEDURES....72

12.1 WINDOWS UPGRADE....72
12.2 UNIX AND LINUX UPGRADE....72
12.3 IBM I UPGRADE....72

13 APPENDIX C – SUPPORT....73

14 APPENDIX D – SUMMARY OF CHANGES....74

15 APPENDIX E – LICENSE AGREEMENT....77

16 APPENDIX F – NOTICES....81

1 Introduction

1.1 Overview

MQ File Mover (MQFM) is a managed file transfer solution that facilitates the transfer of files using IBM's WebSphere MQ (aka MQSeries). MQFM processes “Action” commands which are controlled through an MQFM Workflow XML file. The user combines a series of Action commands to create the MQFM Workflow XML file.

"What is a workflow?" A workflow is a precise definition of how business is conducted or how a business process proceeds. It defines the business process using a precise set of functions - for example, tasks, files, documents, etc.

MQFM is more than a simple file transfer tool, it is a solution for combining business processes into a workflow. The Action commands of a MQFM Workflow XML file are set up by a user to define precisely what actions, aka business processes, are going to be executed, what order they will be executed in and what happens if an error occurs.

MQFM currently contains 26 Action commands. The action commands are in 3 categories: WebShere MQ actions, file actions and other actions.

MQFM's Send and Watch Actions use Advanced Encryption Standard (AES) to encrypt the data and the actions can also compress the data before the data is sent via WebSphere MQ. The Receive Action can decrypt and decompress the incoming data before it is written to the target file.

AES is a data encryption scheme, adopted by the US government, that uses three different key sizes (128-bit, 192-bit, and 256-bit). AES was announced by National Institute of Standards and Technology (NIST) as U.S. FIPS PUB 197 (FIPS 197) on November 26, 2001.

The MQFM Workflow XML Action framework contains 26 Action commands.

WebSphere MQ Actions:

- **Send** - Sends 1 or more files as individual messages via MQ. Each file can be encrypted and/or compressed before it is sent. Each file can be sent to a single queue or multiple queues.
- **Watch** - Monitor for a particular file or monitor a directory for files to appear and then send the file (can be run as a daemon or as a Windows service). Each file can be encrypted and/or compressed before it is sent. Each file can be sent to a single queue or multiple queues.
- **Receive** - Receives an incoming messages and writes them to a file (can be run as a daemon or as a Windows service). Receive Action will decrypt and/or decompress and encrypted and/or compressed messages.
- **PutQuit** - Puts a 'Quit' message on a queue

File Actions:

- **Append** - Appends a file to another file
- **Copy** - Copies 1 or more files from a directory to another directory
- **Delete** - Deletes 1 or more files in a directory
- **DecryptFile** – Decrypt a file using AES 128, 192 or 256-bit decryption
- **EncryptFile** – Encrypt a file using AES 128, 192 or 256-bit encryption
- **Merge** - Merge 2 or more files to another file
- **MergeSort** - Merge 2 or more files and sort the data to another file
- **Move** - Moves 1 or more files from a directory to another directory
- **Rename** - Renames a file
- **ReplaceText** - Performs a search and replace of text in a file.
- **Sort** - Sorts the data of a file into another file
- **Tar** - Combine a file(s) or a directory of files into a tar archive
- **Touch** – Update a file's the modification time or create the file if it does not exist
- **UnTar** – Extract tar archive to a directory
- **UnZip** – Uncompress a zip archive to a directory
- **Zip** - Compresses a file(s) or a directory of files into a Zip archive

Other Actions:

- **Execute** - Runs an external program / application
- **If/Else** - Performs a conditional test against an action's variable
- **Launch** - Invokes an MQFM Workflow XML file.
- **Schedule** - Invokes an MQFM Workflow XML file at a specific date and/or time.
- **SendEmail** – Sends an email to 1 or more recipients.
- **Sleep** – Pause the MQFM Workflow for a period of time.

MQFM can connect to a WebSphere MQ queue manager in 3 possible ways:

- Locally in binding mode
- Remotely using a Client Channel Definition Table (CCDT)
- Remotely using a MQ XML file

MQFM supports both forms of WebSphere MQ security:

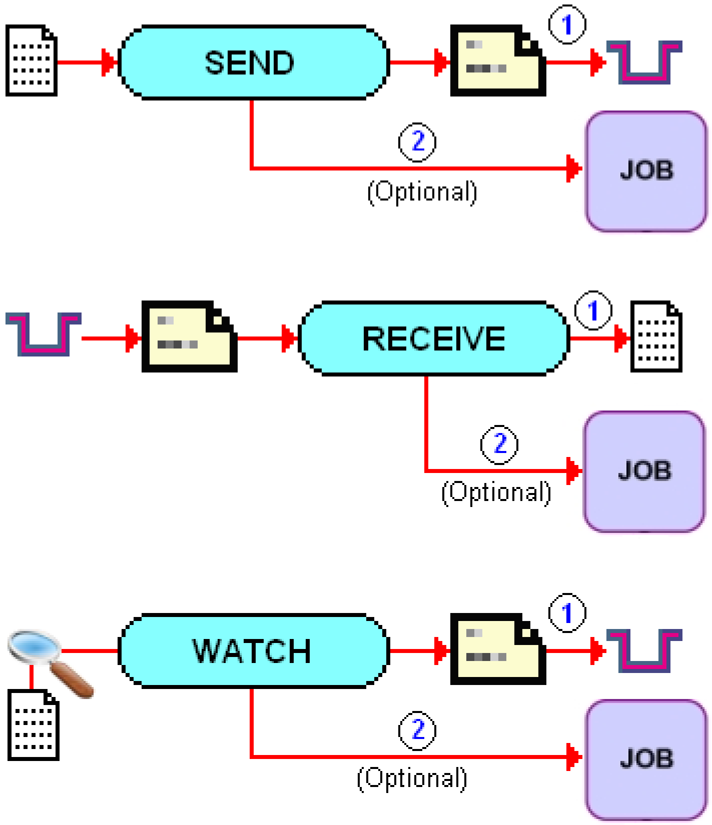
- SSL for connecting to remote queue managers.
- 3rd party security exit for connecting to remote queue managers.

1.2 Executive Summary

The major features of MQFM are as follows:

- A comprehensive set of Action commands used to create a workflow for processing files
- A single Send or Watch Action can send move than 1 file
- A single Receive action can receive move than 1 file
- Assured file delivery using WebSphere MQ
- Send, Watch and Receive Actions use (AES) to encrypt and decrypt the message data
- Send, Watch and Receive Actions support on-the-fly compression and decompression of the message data
- If Action is used to perform a conditional test against an action's variable
- Watch, Receive and Schedule Actions can be run as a Unix/Linux daemon or as a Windows service
- Has file based actions (Append, Copy, Delete, DecryptFile, EncryptFile, Merge, MergeSort, Move, Rename, ReplaceText, Sort, Tar, Touch, UnTar, UnZip and Zip)
- Launch and Schedule Actions can execute other MQFM Workflow XML files
- Ability to launch external programs (Execute)
- Supports both styles of MQ security
- Provides complete logging capability
- Licensed under Apache License 2
- Free to use (support is extra)

1.3 Context Diagram (Logical View)



1.4 Prerequisites

This section provides the minimum supported software levels.

1.4.1 Java

MQ File Mover requires Java v1.5 or higher.

1.4.2 WebSphere MQ

MQ File Mover requires WebSphere MQ v5.3, v6.0 or v7.0 or higher.

1.5 File Definitions

The files used in MQFM are defined as follows:

- ***MQFM_Workflow*** XML file is the file that is defined by a ***MQFM_Workflow.dtd***. It contains the Action commands
- ***MQFM_Email*** XML file is the file that is defined by a ***MQFM_Email.dtd***. It contains information to send an email message via SMTP.
- ***MQFM_Job*** XML file is the file that is defined by a ***MQFM_Job.dtd***. It contains information to run an external program
- ***MQFM_MQ*** XML file is the file that is defined by a ***MQFM_MQ.dtd***. It contains the information to connect to a particular queue manager
- ***MQFM_Schedule*** XML file is the file that is defined by a ***MQFM_Schedule.dtd***. It contains the information for starting an MQFM Workflow XML at a particular date and/or time.
- ***MQFM_Watch*** XML file is the file that is defined by a ***MQFM_Watch.dtd***. It contains the information to monitor a file or files in a directory.
- ***service.properties*** file (*Windows only*) is used to relate a MQFM Windows service name to a MQFM Workflow XML file

2 Installing MQ File Mover

This section describes how to install Capitalware's MQ File Mover.

2.1 Windows Installation

To install MQ File Mover on Windows, do the following instructions:

- Run the install program called: **mqfm-setup.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 File Mover***. This will open a Windows Command prompt.

2.1.1 Installing MQFM as a Windows Service

2.1.1.1 Prerequisites

During the install of WebSphere MQ, the Java MQ JAR files are installed into the <MQ_Install_Path>\java\lib\ directory. E.g. ***C:\Program File\IBM\WebSphere MQ\java\lib\***

The user needs to copy the following Java MQ JAR files to the MQFM "libs" directory (e.g. ***C:\Capitalware\MQFM\libs\***):

1. com.ibm.mq.jar
2. connector.jar

For WebSphere MQ v7.0, the user needs to copy 3 more MQ JAR files to the MQFM "libs" directory :

3. com.ibm.mq.commonservices.jar
4. com.ibm.mq.headers.jar
5. com.ibm.mq.jmqi.jar

2.1.1.2 Installing the MQFM Service

To install MQFM as a Windows service, do the following commands from a Command Prompt:

```
cd \Capitalware\MQFM
mqfm_svc.exe /install
```

The default Windows service name for MQFM is "MQFMService". To install MQFM as many Windows services, do the following commands from a Command Prompt:

```
cd \Capitalware\MQFM
mqfm_svc.exe /install  MQFMService2
mqfm_svc.exe /install  MQFMService3
```

2.1.2 Starting or Stopping the MQFM Service

To start or stop the MQFM Service, the user can use the “net” command from a Command Prompt:

```
net start MQFMService
net stop  MQFMService
```

The user can also start and stop the MQFM Service from the Services MMC console found in the Administrative Tools window.

The default MQFM Workflow XML file that the MQFM Windows service will use is called: “***mqfm.xml***”. To use a differently named MQFM Workflow XML file for the service, edit the ***service.properties*** file and change the MQFM Workflow XML file name.

Example:

```
MQFMService = mqfm_watch.xml
MQFMService2 = mqfm_watch2.xml
```

2.1.3 Uninstalling the MQFM Service

To uninstall MQFM Service, do the following commands from a Command Prompt:

```
cd \Capitalware\MQFM
mqfm_svc.exe /uninstall
```

To uninstall a MQFM Windows service not using the default name, do the following:

```
cd \Capitalware\MQFM
mqfm_svc.exe /uninstall MQFMService2
mqfm_svc.exe /uninstall MQFMService3
```

2.2 Unix and Linux Installation

To install MQ File Mover on Unix or Linux, do the following:

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

```
unzip mqfm.tar.zip
tar -xvf mqfm.tar
```

3. Change directory to ***Capitalware/MQFM/***
4. Next, do the following command:

```
chmod +x *.sh
```

2.3 IBM i Installation

To install MQ File Mover on IBM i (OS/400), do the following:

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

```
unzip mqfm.tar.zip
tar -xvf mqfm.tar
```

3. Change directory to ***Capitalware/MQFM/***
4. Next, do the following command:

```
chmod +x *.sh
```

3 Starting MQFM

This section describes the various ways that the MQFM can be started. Currently, MQFM can be started via WebSphere MQ's triggering system, manually, by a scheduling package or a Windows service.

MQFM supports process locking, so that the user can make sure that only one instance of a MQFM Workflow XML file is being processed at a time (lock_flag of "true") or to make sure another instance of MQFM is already running (lock_flag of "false").

3.1 Command-line parameters

This section describes the required and optional command-line parameters for the MQFM.

Required Parameter	Description
mqfm_workflow.xml	MQFM Workflow XML file to be used for this execution.

Optional Parameter	Description
process_lock_filename	The full path and filename of a file that will be used as a process lock (make it a unique name).
lock_flag	Use either "true" or "false". • "true" means only allow MQFM to run if the process lock filename does not exist • "false" means only allow MQFM to run if the process lock filename does exist

3.2 Triggered Execution

If the local input queue has be setup for triggering (i.e. Trigger First) then when a message arrives on the input queue and the message count goes from 0 to 1 (trigger on first), the queue manager will start MQFM.

3.3 Manual Execution

MQFM can be manually started from a Command Prompt or Unix shell. First change to the MQFM directory (i.e. C:\Capitalware\MQFM) and then type the following:

```
mqfm.bat sample1.xml
```

3.4 Scheduled Execution

MQFM can be setup to be run under a scheduling package. (For example, Windows **Scheduled Task** service or Unix/Linux crontab). Use the scheduling package's tool to setup the necessary parameters for the scheduled execution.

3.5 Windows Service

Please see section 2.1.2 for the details.

3.6 Sample MQFM Workflow XML Files

3.6.1 Sample #1

Sample MQFM Workflow XML file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE MQFM_Workflow SYSTEM "MQFM_Workflow.dtd">
<MQFM_Workflow>
  <Actions>

    <Append file="data\appendtest.txt">
      <File dir="data">test.txt</File>
    </Append>

    <Send delete="Y">
      <File>data\appendtest.txt</File>
      <MQ>
        <QMgrName>MQWT1</QMgrName>
        <QueueName>TEST.Q1</QueueName>
      </MQ>
      <Remote>
        <Directory>C:\temp</Directory>
      </Remote>
    </Send>

    <Send format="S">
      <File>data\abc.txt</File>
      <MQ>
        <QMgrName>MQWT1</QMgrName>
        <QueueName>TEST.Q2</QueueName>
      </MQ>
      <Remote>
        <Directory>C:\temp</Directory>
      </Remote>
    </Send>

  </Actions>
</MQFM_Workflow>
```

This sample file depicts an MQFM Workflow XML file with 3 actions:

1. The Append action appends test.txt file to appendtest.txt
2. The Send action sends file appendtest.txt as a message via MQ and then deletes the file
3. The last Send action sends file abc.txt as a message via MQ and marks the MQMD.Format field as 'MQSTR' (string).

3.6.1.1 Running Sample #1 on Windows

```
mqfm.bat sample1.xml
```

3.6.1.2 Running Sample #1 on Linux/Unix/IBM i

```
mqfm.sh sample1.xml
```

3.6.2 Sample #2

Sample MQFM Workflow XML file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE MQFM_Workflow SYSTEM "MQFM_Workflow.dtd">
<MQFM_Workflow>

  <Global>
    <MQ>
      <QMgrName>MQWT1</QMgrName>
      <QueueName>TEST.Q1</QueueName>
    </MQ>
  </Global>

  <Actions>

    <Copy todir=”D:\appdata”>
      <File dir="data">test.txt</File>
    </Copy>

    <Send delete="Y">
      <File dir=”D:\appdata”>test.txt</File>
      <Remote>
        <Directory>C:\temp</Directory>
      </Remote>
    </Send>

  </Actions>
</MQFM_Workflow>
```

This sample file depicts an MQFM Workflow XML file with a Global section and 2 actions:

Global

- The Global element contains a <MQ> element that holds the MQ values to be used for all actions in this MQFM Workflow XML file.

Actions

1. The Copy action copies test.txt file to D:\appdata\abcxyz.txt
2. The Send action sends file test.txt as a message via MQ and then deletes the file

3.6.2.1 Running Sample #2 on Windows

```
mqfm.bat sample2.xml
```

3.6.2.2 Running Sample #2 on Linux/Unix/IBM i

```
mqfm.sh sample2.xml
```

3.6.3 Sample #3

Sample MQFM Workflow XML file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE MQFM_Workflow SYSTEM "MQFM_Workflow.dtd">
<MQFM_Workflow>

  <Global>
    <MQ>
      <MQFile>mq_mqwt1.xml</MQFile>
    </MQ>
  </Global>

  <Actions>

    <Move tofile=”abcxyz.txt” todir=”D:\appdata”>
      <File dir="data">test.txt</File>
    </Move>

    <Send delete="Y">
      <File dir=”D:\appdata”>abcxyz.txt</File>
      <Remote>
        <Directory>C:\temp</Directory>
      </Remote>
    </Send>

  </Actions>
</MQFM_Workflow>
```

This sample file depicts an MQFM Workflow XML file with a Global section and 2 actions:

Global

- The Global element contains a <MQ> element which contains a <MQFile> element. The <MQFile> contains the name of a MQ XML file that holds the MQ values to be used for all actions in this MQFM Workflow XML file.

Actions

1. The Move action moves the test.txt file to D:\appdata\abcxyz.txt
2. The Send action sends file abcxyz.txt as a message via MQ and then deletes the file

3.6.3.1 Running Sample #3 on Windows

```
mqfm.bat sample3.xml
```

3.6.3.2 Running Sample #3 on Linux/Unix/IBM i

```
mqfm.sh sample3.xml
```

3.6.4 Sample #4

Sample MQFM Workflow XML file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE MQFM_Workflow SYSTEM "MQFM_Workflow.dtd">
<MQFM_Workflow>

  <Global>
    <MQ>
      <CCDTFile>C:\tables\MQWT1.TAB</CCDTFile>
      <QMgrName>MQWT1</QMgrName>
      <QueueName>TEST.Q1</QueueName>
    </MQ>
  </Global>

  <Actions>

    <Send format="S">
      <File dir=”D:\appdata”>abc*</File>
      <Remote>
        <Directory>C:\temp</Directory>
      </Remote>
    </Send>

    <Execute xmlfile="appjob.xml" />

  </Actions>
</MQFM_Workflow>
```

This sample file depicts an MQFM Workflow XML file with a Global section and 2 actions:

Global

- The Global element contains a <MQ> element that has a <CCDTFile> element. The <CCDTFile> value is the complete path and filename of a CCDT (Client Channel Definition Table) file.

Actions

1. The Send action sends all files that begin with abc* as individual messages via MQ and marks the MQMD.Format field as 'MQSTR' (string).
2. The Execute runs an external program as given in the “appjob.xml” file.

3.6.4.1 Running Sample #4 on Windows

```
mqfm.bat sample4.xml
```

3.6.4.2 Running Sample #4 on Linux/Unix/IBM i

```
mqfm.sh sample4.xml
```

3.6.5 Sample #5

Sample MQFM Workflow XML file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE MQFM_Workflow SYSTEM "MQFM_Workflow.dtd">
<MQFM_Workflow>

  <Global>
    <Property name="app_dir" value="D:\appdata" />
    <Property name="myfile" value="test.txt" />
    <MQ>
      <MQFile>mq_mqwt1.xml</MQFile>
    </MQ>
  </Global>

  <Actions>
    <Move todir=”${app_dir}”>
      <File dir="data">${myfile}</File>
    </Move>

    <Send format="S">
      <File dir=”${app_dir}”>${myfile}</File>
      <Remote>
        <Directory>C:\temp</Directory>
      </Remote>
    </Send>

    <Execute xmlfile="appjob.xml" />

    <Delete>
      <File dir="${app_dir}">${myfile}</File>
    </Delete>

  </Actions>
</MQFM_Workflow>
```

This sample file depicts an MQFM Workflow XML file with a Global section and 4 actions:

Global

- The Global element contains 3 different user-defined Property tokens followed by a <MQ> element which contains an <MQFile> element.

Actions

1. The Move action moves a file from one directory to another directory.
2. The Send action sends a file via MQ and marks the MQMD.Format field as 'MQSTR'.
3. The Execute action runs an external program as given in the “appjob.xml” file.
4. The Delete action deletes the specified file.

3.6.5.1 Running Sample #5 on Windows

```
mqfm.bat sample5.xml
```

3.6.5.2 Running Sample #5 on Linux/Unix/IBM i

```
mqfm.sh sample5.xml
```

4 MQFM_Workflow XML File

The MQFM Workflow XML file is a scripting workflow XML file. It is comprised of 2 major elements: Global and Actions. The Global section defines global values to be used while MQFM is running. The Actions section contains Action commands that manipulate files (Append, Copy, Delete, Move, Rename and Zip) and Action commands that move files in and out of MQ (Send, Watch, Receive and PutQuit).

4.1 <MQFM_Workflow> Root Element

<MQFM_Workflow> is the root element for the MQFM Workflow XML file.

4.2 <Global> Element

The Global element (optional) contains 4 elements: Property, LogFile, OnError and MQ.

4.2.1 Attributes

- **onerrorfail** – [Y / N] – specifies whether or not the Workflow should abnormally terminate if an Action fails. The default value is 'Y'. This is a global value that can be overridden for each Action.
- **usefilelocking** – [Y / N] – specifies whether or not the file locking will be used with the actions in this Workflow. The default value is 'Y'. This is a global value.

4.2.2 <Property> Element

The Property element(s) (optional) specifies a used-defined token (keyword) and its value. After the property token is defined, it can be used in Action commands as a replaceable token.

Attributes:

- **name** – The name of the user-defined token
- **value** – The value for the user-defined token

Elements:

None.

Example: Two user-defined properties along with a Send Action.

```
<Global onerrorfail="Y">
  <Property name="testfile" value="abc.txt" />
  <Property name="temp" value="/tmp" />
</Global>
<Actions>
  <Send format="S">
    <File dir="${temp}">${testfile}</File>
    <MQ>
      <QMgrName>MQWT1</QMgrName>
      <QueueName>TEST.Q1</QueueName>
    </MQ>
    <Remote>
      <Directory>C:\temp</Directory>
    </Remote>
  </Send>
</Actions>
```

Example: A user-defined Property can have a value that is made up of other MQFM tokens.

```
<Global>
  <Property name="testfile" value="abc_${DATE}.txt" />
</Global>
```

Example: A user-defined Property can be used within an Execute Action.

```
<Global>
  <Property name="mymsg"
            value="${DATETIME} There is a problem. Help!" />
</Global>
<Actions>
  <Execute xmlfile="abcrun.xml" />
</Actions>
```

And in the ***abcrun.xml*** Job XML file, the file's contents could be as follows:

```
<MQFM_Job>
  <Job name="abcrun">
    <Command wait='y'>C:\pgms\sendalert.exe</Command>
    <Parm>-m</Parm>
    <Parm>${HOSTNAME} - ${mymsg}</Parm>
  </Job>
</MQFM_Job>
```

4.2.3 <LogFile> Element

The LogFile element (optional) specifies the path and filename of the log4j logfile.

```
<Global>
  <LogFile>C:\logs\applA.log</LogFile>
</Global>
```

4.2.4 <OnError> Element

The OnError element (optional) contains 2 element: Execute and SendEmail. OnError can contain either Execute or SendEmail Actions or both.

- **<Execute>** (optional) specifies an external program to be executed when the MQFM Workflow XML script fails.
- **<SendEmail>** (optional) specifies that an email is to be sent when the MQFM Workflow XML script fails.

```
<Global>
  <OnError>
    <Execute xmlfile="alert.xml" />
  </OnError>
</Global>
```

4.2.5 <MQ> Element

The MQ element (optional) contains elements that describe how the action connects to the queue manager.

- **<MQFile>** (optional) specifies a MQ XML file
- **<CCDTFile>** (optional) specifies a CCDT file
- **<QMgrName>** (optional) specifies the name of the queue manager
- **<QueueName>** (optional) specifies the name of the queue
  Attributes of <QueueName>:
  - "**qmgrname**" (optional) specifies the name of a remote queue manager
    e.g. <QueueName qmgrname="MQC1">TEST.Q1</QueueName>
- **<BackOutQName>** (optional) specifies the name of the backout queue name

```
<Global>
  <MQ>
    <MQFile>mq.xml</MQFile>
  </MQ>
</Global>
```

4.3 <Actions> Elements

<Actions> element is a major element in the MQFM Workflow XML file and occurs only once. It has attributes and elements as detailed below.

4.3.1 <Send> Element

This section describes how to invoke the Send Action. The Send Action will read the contents of a file and place it on a queue as a message. The Send Action will include Path and Filename information in the message's MQMD. This information details the file's location on the remote system. The Send Action can handle a file of any size and is sent as one or more messages to a queue. Send Action supports both string and binary message formats. The Send Action can also specify an ***Execute*** that will be executed on the remote server. The Send Action supports a wildcard (i.e. *.txt) in the filename that will send multiple files as separate messages in a single execution. The Send Action can compress and/or encrypt the message data.. Once the data is compressed and/or encrypted, there is no conversion between different platforms (i.e. ASCII to EBCDIC). The Send Action can launch a local external program after putting the message on the queue (see section 6 for more details). It can also delete or archive the file after the message has been put on the queue. The Send Action can send the same message to multiple queues using the DistributionList element in the MQFM_MQ XML file.

Note: US export restrictions limit the Java AES support to 128-bits. If the user wishes to use AES 192 or 256-bit encryption then ***"Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 6"*** needs to be downloaded (and installed) from Oracle's Java web page: http://www.oracle.com/technetwork/java/javase/downloads/index.html

4.3.1.1 Attributes

- **delete** – [Y / N] – deletes the files after it is sent
- **format** – [N / S / Z] – sets the messages MQMD.Format field to None (N), String (S) or compresses (Z) the message data.
- **priority** – [-1 / 0 / 1 / 2 / 3 / 4 / 5 / 6 / 7 / 8 / 9] – sets the priority of the message
- **persistence** – [Y / N] – sets whether or not the message should be persistent
- **keysize** – [128 / 192 / 256] (optional) – specifies the AES key size used for the encryption / decryption of the message data. The default value is 128.
- **passphrase** (optional) – specifies a user supplied PassPhrase.
  Suggested PassPhrase sizes:
    - For KeySize of 128-bits, the PassPhrase should be 16 bytes
    - For KeySize of 192-bits, the PassPhrase should be 24 bytes
    - For KeySize of 256-bits, the PassPhrase should be 32 bytes
- **label** – specifies a label name for the action. The default value is 'default'.
- **onerrorfail** – [Y / N] – specifies whether or not the Workflow should abnormally terminate if the Action fails. The default value is 'Y'.

```
<Send delete="N" format="N" persistence="Y" priority="-1">

<Send delete="N" format="N" persistence="Y" priority="-1"
keysize="128" passphrase="this is a secret">
```

4.3.1.2 Elements

- **<File>** (required) specifies the name of the file to be sent
  Attributes:
    - “dir” - specifies the directory where the file is located
  e.g. <File dir="C:\test">data.txt</File>

- **<MQ>** (required) element contains elements that describe how the Action (Send, Watch, Receive or PutQuit) is to connect to the queue manager.
    - **<MQFile>** (optional) specifies a MQ XML file
    - **<CCDTFile>** (optional) specifies a CCDT file
    - **<QMgrName>** (optional) specifies the name of the queue manager
    - **<QueueName>** (optional) specifies the name of the queue
      Attributes of <QueueName>:
        - "**qmgrname**" (optional) specifies the name of a remote queue manager
          e.g. <QueueName qmgrname="MQC1">TEST.Q1</QueueName>

- **<Execute>** (optional) specifies a local external program to be executed

- **<Archive>** (optional) specifies the directory and filename where the file will be moved to after the file is sent
  Attributes:

    - "todir" (optional) specifies the directory where the file is located
    - "tofile" (optional) specifies the name of the archive file

    e.g. < Archive todir="C:\test" tofile="saved.txt" />

- **<Remote>** (optional) element contains elements that will specify values to be used by the receiving component of MQFM
    - **<Execute>** (optional) specifies a remote external program to be executed
    - **<Directory>** (optional) specifies the remote Job XML file that defines the external program to be executed
    - **<FileName>** (optional) specifies the filename to be used on the remote server

4.3.1.3 Example

An example of a Send Action:

```
<Send delete="N" format="N">
  <File>data\test.xml</File>
  <MQ>
    <QMgrName>MQWT1</QMgrName>
    <QueueName>TEST.Q1</QueueName>
  </MQ>
  <Remote>
    <Directory>C:\temp</Directory>
  </Remote>
</Send>
```

4.3.2 <Watch> Element

This section describes how to invoke the Watch Action. The Watch Action will monitor for a particular file or monitor a directory for a file(s) to appear. The Watch Action can be run in 1 of 3 modes: triggered, as a daemon or as a Windows service The Watch Action will read the contents of a file and place it on a queue as a message. The Watch Action will include Path and Filename information in the message's MQMD. This information details the file's location on the remote system. The Watch Action can handle a file of any size and is sent as one or more messages to a queue. Watch Action supports both string and binary message formats. The Watch Action can also specify an ***Execute*** that will be executed on the remote server. The Watch Action supports a wildcard (i.e. *.txt) in the filename that will send multiple files as separate messages in a single execution. The Watch Action can compress and/or encrypt the message data.. Once the data is compressed and/or encrypted, there is no conversion between different platforms (i.e. ASCII to EBCDIC). The Watch Action can launch a local external program after putting the message on the queue (see section 9 for more details). It can also delete or archive the file after the message has been put on the queue. Watch Action can send the same message to multiple queues using the DistributionList element in the MQFM_MQ XML file.

Note: US export restrictions limit the Java AES support to 128-bits. If the user wishes to use AES 192 or 256-bit encryption then ***"Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 6"*** needs to be downloaded (and installed) from Oracle's Java web page: http://www.oracle.com/technetwork/java/javase/downloads/index.html

4.3.2.1 Attributes

- **xmlfile** (required) the name of the Watch XML file which is located in the ***{MQFM_Install_Path}\watch\*** directory.
- **delete** – [Y / N] – deletes the files after it is sent
- **format** – [N / S / Z] – sets the messages MQMD.Format field to None (N), String (S) or compresses (Z) the message data.
- **priority** – [-1 / 0 / 1 / 2 / 3 / 4 / 5 / 6 / 7 / 8 / 9] – sets the priority of the message
- **persistence** – [Y / N] – sets whether or not the message should be persistent
- **keysize** – [128 / 192 / 256] (optional) – specifies the AES key size used for the encryption / decryption of the message data. The default value is 128.
- **passphrase** (optional) – specifies a user supplied PassPhrase.
  Suggested PassPhrase sizes:
  - For KeySize of 128-bits, the PassPhrase should be 16 bytes
  - For KeySize of 192-bits, the PassPhrase should be 24 bytes
  - For KeySize of 256-bits, the PassPhrase should be 32 bytes
- **label** – specifies a label name for the action. The default value is 'default'.
- **onerrorfail** – [Y / N] – specifies whether or not the Workflow should abnormally terminate if the Action fails. The default value is 'N'.

```
<Watch xmlfile="test_watch.xml" delete="N" format="N" persistence="Y">

<Watch xmlfile="test_watch.xml" delete="N" format="N" persistence="Y"
keysize="128" passphrase="this is a secret">
```

4.3.2.2 Elements

- **<MQ>** (required) element contains elements that describe how the action is to connect to the queue manager.
  - **<MQFile>** (optional) specifies a MQ XML file
  - **<CCDTFile>** (optional) specifies a CCDT file
  - **<QMgrName>** (optional) specifies the name of the queue manager
  - **<QueueName>** (optional) specifies the name of the queue
    Attributes of <QueueName>:
    - "**qmgrname**" (optional) specifies the name of a remote queue manager
      e.g. <QueueName qmgrname="MQC1">TEST.Q1</QueueName>

- **<Execute>** (optional) specifies a local external program to be executed

- **<Archive>** (optional) specifies the directory and filename where the file will be moved to after the file is sent
  Attributes:
  - “todir” (optional) specifies the directory where the file is located
  - “tofile” (optional) specifies the name of the archive file

  e.g. < Archive todir=”C:\test” tofile=”saved.txt” />

- **<Remote>** (optional) element contains elements that will specify values to be used by the receiving component of MQFM
    - **<Execute>** (optional) specifies a remote external program to be executed
    - **<Directory>** (optional) specifies the remote Job XML file that defines the external program to be executed
    - **<FileName>** (optional) specifies the filename to be used on the remote server

4.3.2.3 Example

An example of a Watch Action:

```
<Watch xmlfile="test_watch.xml" delete="N" format="N">
  <MQ>
    <QMgrName>MQWT1</QMgrName>
    <QueueName>TEST.Q1</QueueName>
  </MQ>
  <Remote>
    <Directory>C:\temp</Directory>
  </Remote>
</Watch>
```

4.3.3 <Receive> Element

This section describes how to invoke the Receive Action. The Receive Action will retrieve a message from a queue and write it to a file. The Receive Action can be run in 1 of 4 modes: one-time, triggered, as a daemon or as a Windows service. The Receive Action retrieves the message and writes it to a file under a Unit of Work (UOW).

If the incoming message is not a MQFM message and the user did not specify the Default File name, the Receive Action will use the following format for the file name:
MQFM_####.txt (where #### is the message count number).

If the Receive Action detects a compressed and/or encrypted incoming message, it will decompress and/or decrypt the message before writing it to the output file.

If the incoming message has a specified ***Execute***, the Receive Action will launch an external program after retrieving the message from the queue (see section 6 for more details). The user can assign a default Execute to the Receive Action, so that an external program will be launched when a message is retrieved.

If the Receive Action has a problem with a message (invalid filename or invalid directory), the message will be backed out and the Receive Action will abnormally terminate. In order to avoid termination when a problem message is encountered, the user needs to specify a backout queue name (***BackOutQName***). As a result, the Receive Action will move the problem message to the backout queue and process the next message.

For the Receive Action to exit gracefully while running as a daemon, the PutQuit component is used to place a 'Quit' message on the queue.

Note: US export restrictions limit the Java AES support to 128-bits. If the user wishes to use AES 192 or 256-bit encryption then ***"Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 6"*** needs to be downloaded (and installed) from Oracle's Java web page: http://www.oracle.com/technetwork/java/javase/downloads/index.html

4.3.3.1 Attributes

- **getwithconvert** – [Y / N] – If "Y", the Receive Action will issue a MQGet API call with convert option enabled
- **run** – [S / E / D] – specifies 1 of 3 modes: Single "S", Empty "E" and Daemon "D".
- **keysize** – [128 / 192 / 256] (optional) – specifies the AES key size used for the encryption / decryption of the message data. The default value is 128.
- **passphrase** (optional) – specifies a user supplied PassPhrase.
  Suggested PassPhrase sizes:
    - For KeySize of 128-bits, the PassPhrase should be 16 bytes
    - For KeySize of 192-bits, the PassPhrase should be 24 bytes
    - For KeySize of 256-bits, the PassPhrase should be 32 bytes
- **createdir** – [Y / N] – specifies whether or not the target directory should be created if no such directory exists. The default value is 'N'.

- **label** – specifies a label name for the action. The default value is 'default'.
- **onerrorfail** – [Y / N] – specifies whether or not the Workflow should abnormally terminate if the Action fails. The default value is 'Y'.

```
<Receive getwithconvert="N" run="E">

<Receive getwithconvert="N" run="E" keysize="128" passphrase="this
is a secret">
```

4.3.3.2 Elements

- **<MQ>** (required) element contains elements that describe how the action is to connect to the queue manager.
  - **<MQFile>** (optional) specifies a MQ XML file
  - **<CCDTFile>** (optional) specifies a CCDT file
  - **<QMgrName>** (optional) specifies the name of the queue manager
  - **<QueueName>** (optional) specifies the name of the queue
  - **<BackOutQName>** (optional) specifies the name of the backout queue name

- **<Default>** (optional) element contains elements that will specify values to be used by the receiving component of MQFM
  - **<Execute>** (optional) specifies a local external program to be executed
  - **<Directory>** (optional) specifies the local Job XML file that defines the external program to be executed
  - **<FileName>** (optional) specifies the local filename to be used

4.3.3.3 Example

An example of a Receive Action:

```
<Receive getwithconvert="N" run="E">
  <MQ>
    <QMgrName>MQWT1</QMgrName>
    <QueueName>TEST.Q1</QueueName>
    <BackOutQName>TEST.Q1.BKOUT</BackOutQName>
  </MQ>
</Receive>
```

4.3.4 <PutQuit> Element

This section describes how to invoke the PutQuit Action. The PutQuit Action is used to place a special 'QUIT' message on a queue. PutQuit is used to stop the Receive Action component when it is running as a daemon.

4.3.4.1 Attributes

- **label** – specifies a label name for the action. The default value is 'default'.
- **onerrorfail** – [Y / N] – specifies whether or not the Workflow should abnormally terminate if the Action fails. The default value is 'Y'.

4.3.4.2 Elements

- **<MQ>** (required) element contains elements that describe how the action is to connect to the queue manager.
  - **<MQFile>** (optional) specifies a MQ XML file
  - **<CCDTFile>** (optional) specifies a CCDT file
  - **<QMgrName>** (optional) specifies the name of the queue manager
  - **<QueueName>** (optional) specifies the name of the queue
    Attributes of <QueueName>:
    - "**qmgrname**" (optional) specifies the name of a remote queue manager
      e.g. <QueueName qmgrname="MQC1">TEST.Q1</QueueName>

4.3.4.3 Example

An example of a PutQuit Action:

```
<PutQuit>
  <MQ>
    <QMgrName>MQWT1</QMgrName>
    <QueueName>TEST.Q1</QueueName>
  </MQ>
</PutQuit>
```

4.3.5 <Execute> Element

This section describes how to invoke the Execute Action. The Execute Action is used to run an external program.

4.3.5.1 Attributes

- **xmlfile** (required) specifies the name of the Job XML file (see section 6 for more details)
- **label** – specifies a label name for the action. The default value is 'default'.
- **onerrorfail** – [Y / N] – specifies whether or not the Workflow should abnormally terminate if the Action fails. The default value is 'Y'.

4.3.5.2 Elements

None

4.3.5.3 Example

An example of an Execute Action:

```
<Execute xmlfile="abcrun.xml" />
```

4.3.6 <If> Element

This section describes how to invoke the If and If/Else Action. The If Action is used to perform a conditional test against an action's variable.

4.3.6.1 Attributes

- **labelname** (optional) – specifies the label name of the action for the conditional test. The default value is “default”.
- **actionname** – specifies the action name for the conditional test.
- **fieldname** – specifies the field name of the action for the conditional test.
- **operator** – specifies the type of conditional test to be performed. Valid values are: EQ, NE, LT, LE, GT or GE
- **value** – specifies the value to be used in the conditional test.
- **label** – specifies a label name for the action. The default value is 'default'.
- **onerrorfail** – [Y / N] – specifies whether or not the Workflow should abnormally terminate if the Action fails. The default value is 'Y'.

4.3.6.2 Elements

- For the If section, it can contain any valid Action command
- **<Else>** (optional) element can contain any valid action command

  Else Attributes:
  - **exitrc** (optional) – specifies an exit return code for the else part of the conditional test
  - **onerrorfail** – [Y / N] – specifies whether or not the Workflow should abnormally terminate if the Action fails. The default value is 'Y'.

4.3.6.3 Field Names that can be Tested

The following table shows the field names of each action that can be tested. Note: ***Field names are case sensitive.***

Action Name	Field name	Field Type	Description
Append	fileCount	Integer	The number of files processed.
	srcFilename	String	The source file name.
	toFilename	String	The target file name.
Copy	fileCount	Integer	The number of files processed.
	srcFilename	String	The source file name.
	toFilename	String	The target file name.
	targetDir	String	The full path to the target directory.
DecryptFile	fileCount	Integer	The number of files processed.
	srcFilename	String	The source file name.
	toFilename	String	The target file name.
EncryptFile	fileCount	Integer	The number of files processed.
	srcFilename	String	The source file name.
	toFilename	String	The target file name.
Execute	xmlFile	String	The name of the Job XML file.
Launch	xmlFile	String	The name of the MQFM Workflow XML file.
Merge	fileCount	Integer	The number of files processed.
	srcFilenames	String	The names of the source files.
	toFilename	String	The target file name.
MergeSort	fileCount	Integer	The number of files processed.
	srcFilenames	String	The names of the source files.
	toFilename	String	The target file name.
	column	Integer	The column number to begin the sort
	order	Character	The sort order (A or D)
Move	fileCount	Integer	The number of files processed.
	srcFilename	String	The source file name.
	toFilename	String	The target file name.
	targetDir	String	The full path to the target directory.
PutQuit	CCDTFile	String	The path and name of the CCDT file.
	fileCount	Integer	The number of files processed.

Action Name	Field name	Field Type	Description
	msgCount	Integer	The number of messages processed.
	mqXMLFile	String	The name of the MQ XML file.
	qMgrName	String	The name of the queue manager.
	qName	String	The name of the queue
Receive	backOutQName	String	The name of the backout queue
	CCDTFile	String	The path and name of the CCDT file.
	defaultDirectory	String	The default directory to place the files into
	defaultFilename	String	The default file name for the incoming file
	defaultJobXmlFile	String	The default name of the Job XML file
	fileCount	Integer	The number of files processed.
	msgCount	Integer	The number of messages processed.
	mqXMLFile	String	The name of the MQ XML file.
	overrideDirectoryName	Boolean	Override the incoming directory name?
	overrideFileName	Boolean	Override the incoming file name?
	overrideJobName	Boolean	Override the incoming Job XML file name?
	processType	Boolean	Type of process that the Receive action is running [D/T/E/S]
	qMgrName	String	The name of the queue manager.
	qName	String	The name of the queue
Rename	fileCount	Integer	The number of files processed.
	srcFilename	String	The source file name.
	toFilename	String	The target file name.
MergeSort	fileCount	Integer	The number of files processed.
	srcFilenames	String	The names of the source files.
	toFilename	String	The target file name.
	findText	String	The text to be used for the search
	replaceText	String	The text to be used as the replacement text
Schedule	xmlFile	String	The name of the MQFM Workflow XML file.
SendEmail	msgCount	Integer	The number of messages processed.
	xmlFile	String	The name of the MQFM Workflow XML file.
Send	archiveDirectory	String	The name of the archive directory

Action Name	Field name	Field Type	Description
	archiveFile	Boolean	A flag to signal if the file should be archived
	archiveFileName	String	The name to be given for the archive file
	backOutQName	String	The name of the backout queue
	CCDTFile	String	The path and name of the CCDT file.
	deleteFile	Boolean	Should the file be deleted after processing
	fileCount	Integer	The number of files processed.
	format	Character	Message format
	jobXmlFile	String	The name of the Job XML file
	msgCount	Integer	The number of messages processed.
	mqXMLFile	String	The name of the MQ XML file.
	overrideDirectoryName	Boolean	Override the incoming directory name?
	overrideFileName	Boolean	Override the incoming file name?
	overrideJobName	Boolean	Override the incoming Job XML file name?
	persistence	Boolean	Is the message persistent or not
	priority	Integer	Priority value for the message [0-9]
	remoteDirectory	String	The name of the directory on the remote server
	remoteFilename	String	The name of the file to be used on the remote server
	remoteJobXmlFile	String	The Job XML file to be executed on the remote server
	qMgrName	String	The name of the queue manager.
	qName	String	The name of the queue
	srcFilename	String	The source file name.
Sleep	hours	Integer	The number of hours to sleep for
	minutes	Integer	The number of minutes to sleep for
	seconds	Integer	The number of seconds to sleep for
	milliseconds	Integer	The number of milliseconds to sleep for
Sort	fileCount	Integer	The number of files processed.
	srcFilename	String	The source file name.
	toFilename	String	The target file name.
	column	Integer	The column number to begin the sort

Action Name	Field name	Field Type	Description
	order	Character	The sort order (A or D)
Tar	fileCount	Integer	The number of files processed.
	srcFilename	String	The source file name.
	tarFilename	String	The target tar file name.
Touch	filenames	String	The file names to be touched
UnTar	fileCount	Integer	The number of files processed.
	tarFilename	String	The source tar file name.
	targetDir	String	The full path to the target directory.
UnZip	fileCount	Integer	The number of files processed.
	zipFilename	String	The source zip file name.
	targetDir	String	The full path to the target directory.
Watch	archiveDirectory	String	The name of the archive directory
	archiveFile	Boolean	A flag to signal if the file should be archived
	archiveFileName	String	The name to be given for the archive file
	backOutQName	String	The name of the backout queue
	CCDTFile	String	The path and name of the CCDT file.
	deleteFile	Boolean	Should the file be deleted after processing
	fileCount	Integer	The number of files processed.
	format	Character	Message format
	jobXmlFile	String	The name of the Job XML file
	msgCount	Integer	The number of messages processed.
	mqXMLFile	String	The name of the MQ XML file.
	overrideDirectoryName	Boolean	Override the incoming directory name?
	overrideFileName	Boolean	Override the incoming file name?
	overrideJobName	Boolean	Override the incoming Job XML file name?
	persistence	Boolean	Is the message persistent or not
	priority	Integer	Priority value for the message [0-9]
	remoteDirectory	String	The name of the directory on the remote server
	remoteFilename	String	The name of the file to be used on the remote server
	remoteJobXmlFile	String	The Job XML file to be executed on the

Action Name	Field name	Field Type	Description
			remote server
	qMgrName	String	The name of the queue manager.
	qName	String	The name of the queue
	srcFilename	String	The source file name.
	xmlFile		The name of the Watch XML file
Zip	fileCount	Integer	The number of files processed.
	srcFilename	String	The source file name.
	zipFilename	String	The target zip file name.

4.3.6.4 Example

An example of an If Action:

```
<If actionname="Receive" fieldname="msgCount" operator="GT" value="0">
  <Touch>
    <File dir="C:\temp">test.xml</File>
  </Touch>
</If>
```

An example of an If/Else Action:

```
<If actionname="Receive" fieldname="msgCount" operator="GT" value="0">
  <Touch>
    <File dir="C:\temp">test.xml</File>
  </Touch>
<Else>
  <Copy todir="C:\temp\mqfm" tofile="${FILE}_${DATE}">
    <File dir="data">test.xml</File>
  </Copy>
</Else>
</If>
```

An example of an If/Else Action with a non-zero exit return code that causes an abnormal exit to :

```
<If actionname="Receive" fieldname="msgCount" operator="GT" value="0">
  <Touch>
    <File dir="C:\temp">test.xml</File>
  </Touch>
<Else exitrc=8 />
</If>
```

An example of an If/Else Action containing another If action:

```
<If actionname="Receive" fieldname="msgCount" operator="GT" value="0">
  <Touch>
    <File dir="C:\temp">test.xml</File>
  </Touch>
  <If actionname="Touch" fieldname="fileCount" operator="GT" value="0">
    <Copy todir="C:\temp\mqfm" tofile="${FILE}_${DATE}">
      <File dir="data">test.xml</File>
    </Copy>
  </If>
<Else exitrc=8 />
</If>
```

4.3.7 <Launch> Element

This section describes how to invoke the Launch Action. The Launch Action is used to invoke an MQFM Workflow XML file.

4.3.7.1 Attributes

- **xmlfile** (required) specifies the name of the MQFM Workflow XML file
- **label** – specifies a label name for the action. The default value is 'default'.
- **onerrorfail** – [Y / N] – specifies whether or not the Workflow should abnormally terminate if the Action fails. The default value is 'Y'.

4.3.7.2 Elements

None.

4.3.7.3 Example

An example of an Launch Action:

```
<Launch xmlfile="mqfm_workflow.xml" />
```

4.3.8 <Schedule> Element

This section describes how to invoke the Schedule Action. The Schedules Action is used to invoke an MQFM Workflow XML file at a specific date and/or time.

4.3.8.1 Attributes

- **xmlfile** (required) specifies the name of the Schedule XML file (see section 8 for more details)
- **label** – specifies a label name for the action. The default value is 'default'.
- **onerrorfail** – [Y / N] – specifies whether or not the Workflow should abnormally terminate if the Action fails. The default value is 'Y'.

4.3.8.2 Elements

None.

4.3.8.3 Example

An example of an Schedule Action:

```
<Schedule xmlfile="myschedule.xml" />
```

4.3.9 <SendEmail> Element

This section describes how to invoke the SendEmail Action. The SendEmail Action is used to send an email to 1 or more recipients.

4.3.9.1 Attributes

- **xmlfile** (required) specifies the name of the Email XML file (see section 5 for more details)
- **label** – specifies a label name for the action. The default value is 'default'.
- **onerrorfail** – [Y / N] – specifies whether or not the Workflow should abnormally terminate if the Action fails. The default value is 'Y'.

4.3.9.2 Elements

None

4.3.9.3 Example

An example of an SendEmail Action:

```
<SendEmail xmlfile="sendemail.xml" />
```

4.3.10 <Sleep> Element

This section describes how to invoke the Sleep Action. The Sleep Action is used to pause the Workflow for a period of time.

4.3.10.1 Attributes

- **hours** (optional) specifies number of hours to sleep for.
- **minutes** (optional) specifies number of minutes to sleep for.
- **seconds** (optional) specifies number of seconds to sleep for.
- **milliseconds** (optional) specifies number of milliseconds to sleep for.
- **label** – specifies a label name for the action. The default value is 'default'.
- **onerrorfail** – [Y / N] – specifies whether or not the Workflow should abnormally terminate if the Action fails. The default value is 'Y'.

4.3.10.2 Elements

None

4.3.10.3 Example

An example of an Sleep Action:

```
<Sleep hours="2" minutes="30" seconds="10" />
```

4.3.11 <Append> Element

This section describes how to invoke the Append Action. The Append Action is used to append one file to another file.

4.3.11.1 Attributes

- **file** (required) specifies the path and name of the file to which the data will be appended to.
- **createdir** – [Y / N] – specifies whether or not the target directory should be created if no such directory exists. The default value is 'N'.
- **label** – specifies a label name for the action. The default value is 'default'.
- **onerrorfail** – [Y / N] – specifies whether or not the Workflow should abnormally terminate if the Action fails. The default value is 'Y'.

4.3.11.2 Elements

- **<File>** (required) specifies the name of the source file
  Attributes of <File>:
  - “dir” - specifies the directory where the file is located
  e.g. <File dir=”C:\test”>data.txt</File>

4.3.11.3 Example

An example of an Append Action:

```
<Append file="C:\temp\appendtest.txt">
  <File dir="data">test.xml</File>
</Append>
```

4.3.12 <Copy> Element

This section describes how to invoke the Copy Action. The Copy Action is used to copy one or more files from one directory to another directory.

4.3.12.1 Attributes

- **todir** (required) specifies the target directory where the file is to be copied to
- **tofile** (optional) specifies a new target file name
- **createdir** – [Y / N] – specifies whether or not the target directory should be created if no such directory exists. The default value is 'N'.
- **label** – specifies a label name for the action. The default value is 'default'.
- **onerrorfail** – [Y / N] – specifies whether or not the Workflow should abnormally terminate if the Action fails. The default value is 'Y'.

4.3.12.2 Elements

- **<File>** (required) specifies the name of the file to be copied

  Attributes of <File>:
  - “dir” - specifies the directory where the file is located

  e.g. <File dir=”C:\test”>data.txt</File>

4.3.12.3 Example

An example of a Copy Action:

```
<Copy todir="C:\temp\">
  <File dir="data">test.xml</File>
</Copy>
```

4.3.13 <DecryptFile> Element

This section describes how to invoke the DecryptFile Action. The DecryptFile Action is used to decrypt an encrypted file to another file.

Note: US export restrictions limit the Java AES support to 128-bits. If the user wishes to use AES 192 or 256-bit encryption then ***"Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 6"*** needs to be downloaded (and installed) from Oracle's Java web page: http://www.oracle.com/technetwork/java/javase/downloads/index.html

4.3.13.1 Attributes

- **file** (required) specifies the path and name of the target file
- **keysize** – [128 / 192 / 256] (required) – specifies the AES key size used for the encryption / decryption of the message data. The default value is 128.
- **passphrase** (required) – specifies a user supplied PassPhrase.
  Suggested PassPhrase sizes:
    - For KeySize of 128-bits, the PassPhrase should be 16 bytes
    - For KeySize of 192-bits, the PassPhrase should be 24 bytes
    - For KeySize of 256-bits, the PassPhrase should be 32 bytes
- **createdir** – [Y / N] – specifies whether or not the target directory should be created if no such directory exists. The default value is 'N'.
- **label** – specifies a label name for the action. The default value is 'default'.
- **onerrorfail** – [Y / N] – specifies whether or not the Workflow should abnormally terminate if the Action fails. The default value is 'Y'.

4.3.13.2 Elements

- **<File>** (required) specifies the name of the source/encrypted file
  Attributes of <File>:
    - "dir" - specifies the directory where the file is located
  e.g. <File dir="C:\test">data.txt</File>

4.3.13.3 Example

An example of an DecryptFile Action:

```
<DecryptFile file="C:\temp\test.txt" keysize="128" passphrase="this is a secret">
  <File dir="data">test.enc</File>
</DecryptFile>
```

4.3.14 <Delete> Element

This section describes how to invoke the Delete Action. The Delete Action is used to delete one or more files.

4.3.14.1 Attributes

- **label** – specifies a label name for the action. The default value is 'default'.
- **onerrorfail** – [Y / N] – specifies whether or not the Workflow should abnormally terminate if the Action fails. The default value is 'Y'.

4.3.14.2 Elements

- **<File>** (required) specifies the name of the file to be copied
  Attributes of <File>:
  - “dir” (optional) specifies the directory where the file is located
  e.g. <File dir=”C:\test”>data.txt</File>

4.3.14.3 Example

An example of a Delete Action:

```
<Delete>
  <File dir="data">test.xml</File>
</Delete>
```

Another example of a Delete Action:

```
<Delete>
  <File dir="data">test.xml</File>
  <File dir="data">test2.xml</File>
  <File dir="data">test3.xml</File>
</Delete>
```

4.3.15 <EncryptFile> Element

This section describes how to invoke the EncryptFile Action. The EncryptFile Action is used to encrypt a file to another file.

Note: US export restrictions limit the Java AES support to 128-bits. If the user wishes to use AES 192 or 256-bit encryption then ***"Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 6"*** needs to be downloaded (and installed) from Oracle's Java web page: http://www.oracle.com/technetwork/java/javase/downloads/index.html

4.3.15.1 Attributes

- **file** (required) specifies the path and name of the target/encrypted file
- **keysize** – [128 / 192 / 256] (required) – specifies the AES key size used for the encryption / decryption of the message data. The default value is 128.
- **passphrase** (required) – specifies a user supplied PassPhrase.
  Suggested PassPhrase sizes:
  - For KeySize of 128-bits, the PassPhrase should be 16 bytes
  - For KeySize of 192-bits, the PassPhrase should be 24 bytes
  - For KeySize of 256-bits, the PassPhrase should be 32 bytes
- **createdir** – [Y / N] – specifies whether or not the target directory should be created if no such directory exists. The default value is 'N'.
- **label** – specifies a label name for the action. The default value is 'default'.
- **onerrorfail** – [Y / N] – specifies whether or not the Workflow should abnormally terminate if the Action fails. The default value is 'Y'.

4.3.15.2 Elements

- **<File>** (required) specifies the name of the source file
  Attributes of <File>:
  - "dir" - specifies the directory where the file is located
  e.g. <File dir="C:\test">data.txt</File>

4.3.15.3 Example

An example of an EncryptFile Action:

```
<EncryptFile file="C:\temp\test.enc" keysize="128" passphrase="this is a secret">
  <File dir="data">test.txt</File>
</EncryptFile>
```

4.3.16 <Merge> Element

This section describes how to invoke the Merge Action. The Merge Action is used to merge 2 or more files to another file.

4.3.16.1 Attributes

- **file** (required) specifies the path and name of the file to which the data will be merged to.
- **createdir** – [Y / N] – specifies whether or not the target directory should be created if no such directory exists. The default value is 'N'.
- **label** – specifies a label name for the action. The default value is 'default'.
- **onerrorfail** – [Y / N] – specifies whether or not the Workflow should abnormally terminate if the Action fails. The default value is 'Y'.

4.3.16.2 Elements

The File element repeats for each source file.

- **<File>** (required) specifies the name of one of the source file
  Attributes of <File>:
  - “dir” - specifies the directory where the file is located
  e.g. <File dir=”C:\test”>data.txt</File>

4.3.16.3 Example

An example of an Merge Action:

```
<Merge file="C:\temp\mergetest.txt">
  <File dir="data">test.txt</File>
  <File dir="data">test2.txt</File>
  <File dir="data">test3.txt</File>
  <File dir="data">test4.txt</File>
</Merge>
```

4.3.17 <MergeSort> Element

This section describes how to invoke the MergeSort Action. The MergeSort Action is used to merge 2 or more files, sort the data into another file.

4.3.17.1 Attributes

- **file** (required) specifies the path and name of the file to which the data will be merged to.
- **column** (optional) specifies the starting column to sort the data on. The default value is “1”,
- **order** [A/D] (optional) specifies sort order of either ascending or descending. The default value is “A”.
- **createdir** – [Y / N] – specifies whether or not the target directory should be created if no such directory exists. The default value is 'N'.
- **label** – specifies a label name for the action. The default value is 'default'.
- **onerrorfail** – [Y / N] – specifies whether or not the Workflow should abnormally terminate if the Action fails. The default value is 'Y'.

4.3.17.2 Elements

The File element repeats for each source file.

- **<File>** (required) specifies the name of one of the source file
  Attributes of <File>:
  - “dir” - specifies the directory where the file is located
  e.g. <File dir=”C:\test”>data.txt</File>

4.3.17.3 Example

An example of an MergeSort Action:

```
<MergeSort file="C:\temp\mergetest.txt" column="1" order="A">
  <File dir="data">test.txt</File>
  <File dir="data">test2.txt</File>
  <File dir="data">test3.txt</File>
  <File dir="data">test4.txt</File>
</MergeSort>
```

4.3.18 <Move> Element

This section describes how to invoke the Move Action. The Move Action is used to move one or more files from one directory to another directory.

4.3.18.1 Attributes

- **todir** (required) specifies the target directory where the file is to be moved to
- **tofile** (optional) specifies a new target file name
- **createdir** – [Y / N] – specifies whether or not the target directory should be created if no such directory exists. The default value is 'N'.
- **label** – specifies a label name for the action. The default value is 'default'.
- **onerrorfail** – [Y / N] – specifies whether or not the Workflow should abnormally terminate if the Action fails. The default value is 'Y'.

4.3.18.2 Elements

- **<File>** (required) specifies the name of the file to be moved
  Attributes of <File>:
  - “dir” (optional) specifies the directory where the file is located
  e.g. <File dir=”C:\test”>data.txt</File>

4.3.18.3 Example

An example of a Move Action:

```
<Move todir="C:\temp\">
  <File dir="data">test.xml</File>
</Move>
```

4.3.19 <Rename> Element

This section describes how to invoke the Rename Action. The Rename Action is used to rename a file.

4.3.19.1 Attributes

- **tofile** (required) specifies a new file name
- **label** – specifies a label name for the action. The default value is 'default'.
- **onerrorfail** – [Y / N] – specifies whether or not the Workflow should abnormally terminate if the Action fails. The default value is 'Y'.

4.3.19.2 Elements

- **<File>** (required) specifies the name of the file to be moved
  Attributes of <File>:
  - “dir” (optional) specifies the directory where the file is located

  e.g. <File dir=”C:\test”>data.txt</File>

4.3.19.3 Example

An example of a Rename Action:

```
<Rename tofile="test2.xml">
  <File dir="data">test.xml</File>
</Rename>
```

4.3.20 <ReplaceText> Element

This section describes how to invoke the ReplaceText Action. The ReplaceText Action is used to search and replace text in a file.

4.3.20.1 Attributes

- **file** (required) specifies the path and name of the target file
- **rows** (required) specifies the rows to perform the search against. The default is “1:*” which is all rows. The layout of the field is S:E where S is the starting row and E is the ending row (“*” until the end of the file)
- **columns** (required) specifies the columns to perform the search against. The default is “1:*” which is all columns. The layout of the field is S:E where S is the starting column and E is the ending column (“*” up to the last column of data)
- **createdir** – [Y / N] – specifies whether or not the target directory should be created if no such directory exists. The default value is 'N'.
- **label** – specifies a label name for the action. The default value is 'default'.
- **onerrorfail** – [Y / N] – specifies whether or not the Workflow should abnormally terminate if the Action fails. The default value is 'Y'.

4.3.20.2 Elements

- **<File>** (required) specifies the name of the source file
  Attributes of <File>:
  - “dir” - specifies the directory where the file is located
  e.g. <File dir=”C:\test”>data.txt</File>

- **<Find>** (required) specifies the text to be searched for
  e.g. <Find>XXX</Find>

- **<Replace>** (required) specifies the replacement text
  e.g. <Replace>YYY</Replace>

4.3.20.3 Example

An example of an ReplaceText Action:

```
<ReplaceText file="C:\temp\test.txt" rows="1:*" columns="1:*">
  <Find>000</Find>
  <Replace>!!!</Replace>
  <File>data\test.txt</File>
</ReplaceText>
```

4.3.21 <Sort> Element

This section describes how to invoke the Sort Action. The Sort Action is used to sort the data into another file.

4.3.21.1 Attributes

- **file** (required) specifies the path and name of the file to which the data will be merged to.
- **column** (optional) specifies the starting column to sort the data on. The default value is “1”,
- **order** [A/D] (optional) specifies sort order of either ascending or descending. The default value is “A”.
- **createdir** – [Y / N] – specifies whether or not the target directory should be created if no such directory exists. The default value is 'N'.
- **label** – specifies a label name for the action. The default value is 'default'.
- **onerrorfail** – [Y / N] – specifies whether or not the Workflow should abnormally terminate if the Action fails. The default value is 'Y'.

4.3.21.2 Elements

The File element repeats for each source file.

- **<File>** (required) specifies the name of one of the source file
  Attributes of <File>:
  - “dir” - specifies the directory where the file is located
  e.g. <File dir=”C:\test”>data.txt</File>

4.3.21.3 Example

An example of an Sort Action:

```
<Sort file="C:\temp\mergetest.txt" column="1" order="A">
  <File dir="data">test.txt</File>
</Sort>
```

4.3.22 <Tar> Element

This section describes how to invoke the Tar Action. The Tar Action is used to create a tar archive from the contents of a file or files in a directory.

4.3.22.1 Attributes

- **file** (required) specifies a target tar file name
- **createdir** – [Y / N] – specifies whether or not the target directory should be created if no such directory exists. The default value is 'N'.
- **label** – specifies a label name for the action. The default value is 'default'.
- **onerrorfail** – [Y / N] – specifies whether or not the Workflow should abnormally terminate if the Action fails. The default value is 'Y'.

4.3.22.2 Elements

- **<File>** (required) specifies the name of the file to be compressed
  Attributes of <File>:
  - “dir” (optional) specifies the directory where the file is located
  e.g. <File dir=”C:\test”>data.txt</File>

4.3.22.3 Example

An example of a Tar Action:

```
<Tar file="C:\temp\test.tar">
  <File dir="data">test.xml</File>
</Tar>
```

4.3.23 <Touch> Element

This section describes how to invoke the Touch Action. The Touch Action is used to set the modification time of the file to the current time of day. If the file doesn't exist, it is created.

4.3.23.1 Attributes

- **createdir** – [Y / N] – specifies whether or not the target directory should be created if no such directory exists. The default value is 'N'.
- **label** – specifies a label name for the action. The default value is 'default'.
- **onerrorfail** – [Y / N] – specifies whether or not the Workflow should abnormally terminate if the Action fails. The default value is 'Y'.

4.3.23.2 Elements

- **<File>** (required) specifies the name of the source file

  Attributes of <File>:
  - “dir” - specifies the directory where the file is located

  e.g. <File dir=”C:\test”>data.txt</File>

4.3.23.3 Example

An example of an Touch Action:

```
<Touch>
  <File dir="data">test.xml</File>
</Touch>
```

Another example of an Touch Action:

```
<Touch>
  <File dir="data">test.xml</File>
  <File dir="data">test2.xml</File>
  <File dir="data">test3.xml</File>
</Touch>
```

4.3.24 <UnTar> Element

This section describes how to invoke the UnTar Action. The UnTar Action is used to extract the contents of a tar archive to a directory.

4.3.24.1 Attributes

- **todir** (required) specifies a target directory
- **createdir** – [Y / N] – specifies whether or not the target directory should be created if no such directory exists. The default value is 'N'.
- **label** – specifies a label name for the action. The default value is 'default'.
- **onerrorfail** – [Y / N] – specifies whether or not the Workflow should abnormally terminate if the Action fails. The default value is 'Y'.

4.3.24.2 Elements

- **<File>** (required) specifies the name of the tar file
  Attributes of <File>:
  - “dir” (optional) specifies the directory where the file is located
  e.g. <File dir=”C:\test”>data.tar</File>

4.3.24.3 Example

An example of a UnTar Action:

```
<UnTar todir="C:\temp">
  <File dir="data">test.tar</File>
</UnTar>
```

4.3.25 <UnZip> Element

This section describes how to invoke the UnZip Action. The UnZip Action is used to extract the contents of a zip archive to a directory.

4.3.25.1 Attributes

- **todir** (required) specifies a target directory
- **createdir** – [Y / N] – specifies whether or not the target directory should be created if no such directory exists. The default value is 'N'.
- **label** – specifies a label name for the action. The default value is 'default'.
- **onerrorfail** – [Y / N] – specifies whether or not the Workflow should abnormally terminate if the Action fails. The default value is 'Y'.

4.3.25.2 Elements

- **<File>** (required) specifies the name of the zip file
  Attributes of <File>:
  - "dir" (optional) specifies the directory where the file is located
  e.g. <File dir="C:\test">data.zip</File>

4.3.25.3 Example

An example of a Zip Action:

```
<UnZip todir="C:\temp">
  <File dir="data">test.zip</File>
</UnZip>
```

4.3.26 <Zip> Element

This section describes how to invoke the Zip Action. The Zip Action is used to compress the contents of a file or files in a directory.

4.3.26.1 Attributes

- **file** (required) specifies a target zip file name
- **createdir** – [Y / N] – specifies whether or not the target directory should be created if no such directory exists. The default value is 'N'.
- **label** – specifies a label name for the action. The default value is 'default'.
- **onerrorfail** – [Y / N] – specifies whether or not the Workflow should abnormally terminate if the Action fails. The default value is 'Y'.

4.3.26.2 Elements

- **<File>** (required) specifies the name of the file to be compressed
  Attributes of <File>:
  - "dir" (optional) specifies the directory where the file is located
  e.g. <File dir="C:\test">data.txt</File>

4.3.26.3 Example

An example of a Zip Action:

```
<Zip file="C:\temp\test.zip">
  <File dir="data">test.xml</File>
</Zip>
```

5 MQFM_Email XML File

This section describes how to create an Email XML file for use by MQFM SendEmail Action. The Email XML file contains information to send an email message via SMTP.

5.1 <MQFM_Email> Root Element

<MQFM_Email> is the root element for the MQFM_Email XML file. The MQFM_Email root element is comprised of 1 element (Email).

Note: The MQFM_Email XML files must be stored in the <MQFM_Install_PATH>\email\ directory. i.e. C:\Capitalware\MQFM\email\

5.2 <SmtpHost> Element

The SmtpHost element contains the SMTP hostname or IP address.

Attributes:

- "**authentication**" (optional) specifies if the connection will use UserID and Password authentication. If so, then the SmtpUserID and SmtpPassword elements must be specified.
- "**ssl**" (optional) specifies if the connection will use SSL

e.g.
<SmtpHost ssl="Y" authentication="Y">mail.acme.com</SmtpHost>

5.3 <SmtpPort> Element

The SmtpPort element contains the SMTP port number. The default value is 25. Note: Certain SMTP servers require that the user use either 465 or 587 as the port number.

5.4 <SmtpUserID> Element

The SmtpUserID element contains the UserID to be used when authenticating against an SMTP server.

5.5 <SmtpPassword> Element

The SmtpPassword element contains the password to be used when authenticating against an SMTP server.

5.6 <FromEmailAddress> Element

The FromEmailAddress element contains the “From” email address.

5.7 <ToEmailAddress> Element

The ToEmailAddress element contains the “To” email addresses of the email to be sent. Separate each email address with a comma (',').
e.g.
<ToEmailAddress>fred@acme.com,barney@acme.com,pebbles@acme.com</ToEmailAddress>

5.8 <Subject> Element

The Subject element contains the subject of the email to be sent.

5.9 <MessageText> Element

The MessageText element contains the message text (body) of the email to be sent.

5.10 Sample

To create an MQFM_Email XML file, open a text editor and input one or more of the above elements as shown in the example below:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE MQFM_Email SYSTEM "MQFM_Email.dtd">
<MQFM_Email>
  <SmtpHost ssl="Y" authentication="Y">mail.acme.com</SmtpHost>
  <SmtpPort>465</SmtpPort>
  <SmtpUserID>mrslate@acme.com</SmtpUserID>
  <SmtpPassword>mypwd</SmtpPassword>
  <FromEmailAddress>mrslate@acme.com</FromEmailAddress>
  <ToEmailAddress>fred@acme.com,barney@acme.com</ToEmailAddress>
  <Subject>test subject</Subject>
  <MessageText>This is a test message
The second line of the test message
The third line of the test message</MessageText>
</MQFM_Email>
```

Note: The MQFM_Email XML files must be stored in the <MQFM_Install_PATH>\email\ directory. i.e. C:\Capitalware\MQFM\email\

6 MQFM_Job XML File

This section describes how to create an Job XML file for use by MQFM Execute Action. The Job XML file contains configuration information to run an external program.

6.1 <MQFM_Job> Root Element

<MQFM_Job> is the root element for the MQFM_Job XML file. The MQFM_Job root element is comprised of 1 element (Job).

If the user is using a Send Action to send a single file, and invoking a local external command, do not use a Command's ***wait*** attribute with a value of "N", since the main process may terminate before the child completes running the external program.

Note: The MQFM_Job XML files must be stored in the <MQFM_Install_PATH>\jobs\ directory. i.e. C:\Capitalware\MQFM\jobs\

6.2 <Job> Element

The Job element is comprised of a Command element, an optional Directory element and one or more Parm elements.

- **<Command>** (required) specifies the external program to be executed
  Attributes of <Command>:
  - "**wait**" [Y / N] specifies the directory in which the file is located
  e.g. <Command wait="Y">C:\temp\abc.exe</Command>

- **<Directory>** (optional) specifies the working directory for the command

- **<Parm>** (optional) specifies the parameter for the command. Note: Place each parameter in a separate <Parm> element.

6.3 Sample

To create an MQFM_Job XML file, open a text editor and input one or more of the above elements as shown in the example below:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE MQFM_Job SYSTEM "MQFM_Job.dtd">
<MQFM_job>
  <Job>
    <Command wait='N'>C:\test\abc.exe</Command>
    <Directory>C:\test</Directory>
    <Parm>-f</Parm>
    <Parm>data/test.xml</Parm>
    <Parm>-m</Parm>
    <Parm>MQWT1</Parm>
    <Parm>-q</Parm>
    <Parm>TEST.Q10</Parm>
    <Parm>-d</Parm>
    <Parm>C:\Temp</Parm>
  </Job>
</MQFM_job>
```

Note: The MQFM_Job XML files must be stored in the <MQFM_Install_PATH>\jobs\ directory. i.e. C:\Capitalware\MQFM\jobs\

7 MQFM_MQ XML File

This section describes how to create an MQFM_MQ XML file for use by MQFM. The MQFM_MQ XML file contains the MQ configuration information for connecting to a remote queue manager as described below.

7.1 <MQFM_MQ> Root Element

<MQFM_MQ> is the root element for the MQFM_MQ XML file. The MQFM_MQ root element is comprised of 19 elements. According to the user's needs, one or more of the following elements are selected:

- **<QMgrName>** (optional) specifies the name of the queue manager
- **<QueueName>** (optional) specifies the name of the queue
  Attributes of <QueueName>:
  - "**qmgrname**" (optional) specifies the name of a remote queue manager
    e.g. <QueueName qmgrname="MQC1">TEST.Q1</QueueName>

- **<CCDTFile>** (optional) specifies a CCDT file
- **<Hostname>** (optional) specifies the hostname
- **<Port>** (optional) specifies the port number
- **<ChannelName>** (optional) specifies the channel name
- **<UserID>** (optional) specifies the UserID
- **<Password>** (optional) specifies the Password for the UserID
- **<SecurityExit>** (optional) specifies the security exit name
- **<SecurityExitPath>** (optional) specifies the path to the security exit
- **<CipherSuiteName>** (optional) specifies the the CipherSuite name
- **<DistinguishedName>** (optional) specifies the Distinguished name
- **<TrustedStore>** (optional) specifies the Trusted Store
- **<TrustedStorePasswd>** (optional) specifies the Trusted Store password
- **<KeyStore>** (optional) specifies the Key Store
- **<KeyStorePasswd>** (optional) specifies the Key Store password
- **<LDAPServer>** (optional) specifies the LDAP server hostname or IP address
- **<LDAPServerPort>** (optional) specifies the LDAP server port number

- **<DistributionList>** (optional) specifies a list of the queue names
  Elements of <DistributionList>:
  - **<QueueName>** (optional) specifies the name of the queue
  Attributes of <QueueName>:
    - "**qmgrname**" (optional) specifies the name of a remote queue manager
      e.g. <QueueName qmgrname="MQC1">TEST.Q1</QueueName>

7.2 Sample

To create an MQFM_MQ XML file, open a text editor and input one or more of the above elements into the MQFM_MQ XML file.

An example of a standard MQFM_MQ XML file for connecting to a local queue manager and specifying a single queue:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE MQFM_MQ SYSTEM "MQFM_MQ.dtd">
<MQFM_MQ>
    <QMgrName>MQWT1</QMgrName>
    <QueueName>TEST.Q1</QueueName>
    <UserID>abcuid</UserID>
</MQFM_MQ>
```

An example of a standard MQFM_MQ XML file for connecting to a remote queue manager and specifying a single queue:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE MQFM_MQ SYSTEM "MQFM_MQ.dtd">
<MQFM_MQ>
    <QMgrName>MQWT1</QMgrName>
    <QueueName>TEST.Q1</QueueName>
    <Hostname>10.10.10.10</Hostname>
    <ChannelName>SYSTEM.DEF.SVRCONN</ChannelName>
    <Port>1414</Port>
    <UserID>abcuid</UserID>
</MQFM_MQ>
```

An example of a MQFM_MQ XML file for connecting to a remote queue manager and specifying a list of queues that each message will be sent to:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE MQFM_MQ SYSTEM "MQFM_MQ.dtd">
<MQFM_MQ>
    <QMgrName>MQWT1</QMgrName>
    <DistributionList>
      <QueueName>TEST.Q1</QueueName>
      <QueueName>TEST.Q2</QueueName>
      <QueueName>TEST.Q3</QueueName>
    </DistributionList>
    <Hostname>10.10.10.10</Hostname>
    <ChannelName>SYSTEM.DEF.SVRCONN</ChannelName>
    <Port>1414</Port>
    <UserID>abcuid</UserID>
</MQFM_MQ>
```

Note: The MQFM_MQ XML files must be stored in the <MQFM_Install_PATH>\mq\ directory. i.e. C:\Capitalware\MQFM\mq\

8 MQFM_Schedule XML File

This section describes how to create a Schedule XML file for use by MQFM Schedule Action. The Schedule XML file contains information for starting an MQFM Workflow XML at a particular date and/or time.

8.1 <MQFM_Schedule> Root Element

<MQFM_Schedule> is the root element for the MQFM_Schedule XML file.

Note: The MQFM_Schedule XML files must be stored in the <MQFM_Install_PATH>\schedule\ directory. i.e. C:\Capitalware\MQFM\schedule\

8.2 <Schedule> Element

The Schedule element (required) contains the information for starting an MQFM Workflow XML at a particular date and/or time.

8.2.1 Attributes

- **xmlfile** (required) specifies the name of the MQFM Workflow XML file

8.2.2 Elements

- **<Minute>** (optional) specifies the minute(s) for the scheduled event. The default value is "*" which means it will be run once every minute. Valid values are 0-59. The user can input a single minute value (i.e. 0) or a group of minutes (i.e. 0,10,20,30,40,50).

- **<Hour>** (optional) specifies the hour(s) for the scheduled event. The default value is "*". Valid values are 0-23. The user can input a single hour value (i.e. 0) or a group of hours (i.e. 0,6,12,18).

- **<DayOfMonth>** (optional) specifies the DayOfMonth for the scheduled event. The default value is "*". Valid values are 1-31. The user can input a single day value (i.e. 1) or a group of hours (i.e. 0,6,12,18).

- **<Month>** (optional) specifies the hour(s) for the scheduled event. The default value is "*". Valid values are 1-12 (or use the names of months). The user can input a single month value (i.e. 0) or a group of hours (i.e. 1,4,7,10).

- **<DayOfWeek>** (optional) specifies the day(s) for the scheduled event. The default value is "*". Valid values are 0-7 (0 or 7 for Sunday or use the names of days). The user can input a single day value (i.e. 2 or TUESDAY) or a group of hours (i.e. 1,3,5 or MONDAY, WEDNESDAY,FRIDAY).

- **<Year>** (optional) specifies the hour(s) for the scheduled event. The default value is "*". The user can input a single year value (i.e. 2011) or a group of years (i.e. 2011,2013,2015).

8.3 Sample

To create an MQFM_Schedule XML file, open a text editor and input one or more of the above elements as shown in the example below:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE MQFM_Schedule SYSTEM "MQFM_Schedule.dtd">
<MQFM_Schedule>

  <Schedule xmlfile="mqfm_workflow.xml">
    <Minute>*</Minute>
    <Hour>*</Hour>
    <DayOfMonth>*</DayOfMonth>
    <Month>*</Month>
    <DayOfWeek>*</DayOfWeek>
    <Year>*</Year>
  </Schedule>

</MQFM_Schedule>
```

Note: The MQFM_Schedule XML files must be stored in the <MQFM_Install_PATH>\schedule\ directory. i.e. C:\Capitalware\MQFM\schedule\

9 MQFM_Watch XML File

This section describes how to create a Watch XML file for use by MQFM Watch Action. The Watch XML file contains information to monitor for the appearance of a file or files within a directory.

9.1 <MQFM_Watch> Root Element

<MQFM_Watch> is the root element for the MQFM_Watch XML file. The MQFM_Watch root element is comprised of an optional PollInternal element and one or more Watch elements.

Note: The MQFM_MQ XML files must be stored in the <MQFM_Install_PATH>\watch\ directory. i.e. C:\Capitalware\MQFM\watch\

9.2 <PollInterval> Element

<PollInterval> element (optional) contains the polling interval in seconds for this set of Watch elements. The PollInterval value must be larger than 5 seconds.

9.3 <Watch> Element

The Watch element (required) contains the information to monitor the file(s) or files in a directory.

9.3.1 Attributes

- "**type**" [F / D] specifies whether Watch Action is monitoring a file or files within a directory

```
<Watch type="F">
```

9.3.2 Elements

When Watch element's ***type*** attribute has a value of “F”, use the <File> element as follows:

- **<File>** (required) specifies the name of the file to be moved
  Attributes of <File>:
  - “dir” (optional) specifies the directory where the file is located
  e.g. <File dir=”C:\test”>data.txt</File>

When Watch element's ***type*** attribute has a value of “D”, use the <Directory> element with or without the <Extension> element as follows:

- **<Directory>** (required) specifies the complete path to a directory
  Attributes of <Directory>:
  - “sendonstartup” [Y/N] (optional) specifies that MQFM should process matching files in the directory. Otherwise, existing files are ignored and only new files are processed.
  e.g. <Directory sendonstartup="Y">C:\work\test</Directory>

- **<Extension>** (optional) specifies the file extension to monitor in a directory

9.3.2.1 <WatchSend> Element

The <WatchSend> element can be optionally used within the <Watch> element. Any values given in the <WatchSend> element will override the Watch values specified in the <Watch> element of the MQFM Workflow XML file.

Attributes

- **delete** – [Y / N] – deletes the files after it is sent
- **format** – [N / S / Z] – sets the messages MQMD.Format field to None (N), String (S) or compresses (Z) the message data.
- **priority** – [-1 / 0 / 1 / 2 / 3 / 4 / 5 / 6 / 7 / 8 / 9] – sets the priority of the message
- **persistence** – [Y / N] – sets whether or not the message should be persistent
- **onerrorfail** – [Y / N] – specifies whether or not the Workflow should abnormally terminate if the Action fails. The default value is 'N'.

```
<WatchSend delete="N" format="N" persistence="Y" priority="-1">
```

Elements

- **<MQ>** (optional) element contains elements that describe how the Action (Send, Watch, Receive or PutQuit) is to connect to the queue manager.
    - **<MQFile>** (optional) specifies a MQ XML file
    - **<CCDTFile>** (optional) specifies a CCDT file
    - **<QMgrName>** (optional) specifies the name of the queue manager
    - **<QueueName>** (optional) specifies the name of the queue
      Attributes of <QueueName>:
        - "**qmgrname**" (optional) specifies the name of a remote queue manager
          e.g. <QueueName qmgrname="MQC1">TEST.Q1</QueueName>

- **<Execute>** (optional) specifies a local external program to be executed

- **<Archive>** (optional) specifies the directory and filename where the file will be moved to after the file is sent
  Attributes:
    - “todir” (optional) specifies the directory where the file is located
    - “tofile” (optional) specifies the name of the archive file

  e.g. < Archive todir=”C:\test” tofile=”saved.txt” />

- **<Remote>** (optional) element contains elements that will specify values to be used by the receiving component of MQFM
    - **<Execute>** (optional) specifies a remote external program to be executed
    - **<Directory>** (optional) specifies the remote Job XML file that defines the external program to be executed
    - **<FileName>** (optional) specifies the filename to be used on the remote server

9.4 Sample

To create an MQFM_Watch XML file, open a text editor and input one or more of the above elements as shown in the example below:

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE MQFM_Watch SYSTEM "MQFM_Watch.dtd">
<MQFM_Watch>
  <PollInterval>10</PollInterval>

  <Watch type="F">
    <File>C:\temp\mqwt1.log</File>
    <WatchSend>
      <MQ>
        <QMgrName>MQWT1</QMgrName>
        <QueueName>TEST.Q1</QueueName>
      </MQ>
      <Archive todir="C:\temp\MQFM\archive" tofile="${FILE}" />
    </WatchSend>
  </Watch>

  <Watch type="D">
    <Directory>C:\work\test</Directory>
    <Extension>txt</Extension>
    <WatchSend>
      <MQ>
        <QMgrName>MQWT1</QMgrName>
        <QueueName>TEST.Q2</QueueName>
      </MQ>
    </WatchSend>
  </Watch>

</MQFM_Watch>
```

Note: The MQFM_Watch XML files must be stored in the <MQFM_Install_PATH>\watch\ directory. i.e. C:\Capitalware\MQFM\watch\

10 MQFM Tokens

This section describes how token replacement will be done in MQFM.

10.1 Tokens

10.1.1 Date / Time Tokens

The following is a list of the Date and Time tokens that will be replaced:

Token	Description
${DATE}	Date in the format: YYYY_MM_DD e.g. 2009_05_06
${DATETIME}	Date in the format: YYYY_MM_DD_HH_MM_SS.SSS e.g. 2009_04_30_22_10_07.123
${DATEM}	Substract the user-defined Property “${DATEMINUS}” from the current date. Date in the format: YYYY_MM_DD e.g. 2009_05_06
${DATE-1} ${DATE-2} ${DATE-3} ${DATE-4} ${DATE-5} ${DATE-6} ${DATE-7}	Substract the value (i.e. 1) from the current date. Date in the format: YYYY_MM_DD e.g. 2009_05_06
${DATEP}	Add the user-defined Property “${DATEPLUS}” to the current date. Date in the format: YYYY_MM_DD e.g. 2009_05_06
${DATE+1} ${DATE+2} ${DATE+3} ${DATE+4} ${DATE+5} ${DATE+6} ${DATE+7}	Add the value (i.e. 1) to the current date. Date in the format: YYYY_MM_DD e.g. 2009_05_06
${TIME}	Time in the format: HH_MM_SS.SSS e.g. 20_37_55.445
${YYYY}	Year in the format: YYYY e.g. 2009
${MM}	Month in the format: MM e.g. 05
${DD}	Day in the format: DD e.g. 14
${HH}	Hour in the format: HH e.g. 19
${hh}	Round the minutes to the next half hour: hh e.g.00 or 30
${MIN}	Minutes in the format: MM e.g. 47
${SS}	Seconds in the format: SS e.g. 33

10.1.2 File Tokens

The following is a list of the file tokens that will be replaced:

Token	Description
${FILE}	The complete “from” file name e.g. abc.txt
${FILE_PREFIX}	The prefix portion of the file name e.g. abc
${FILE_EXT}	The file extension of the file name e.g. txt

10.1.3 Index Token

The INDEX token is used to add an index counter to the file name:

Token	Description
${INDEX}	Index counter of the file currently being processed e.g. 0003

10.1.4 General Tokens

The following is a list of general tokens that can be replaced:

Token	Description
${ACTION}	The name of the Action component
${HOSTNAME}	The hostname of the server that MQFM is running on
${IPADDRESS}	The IP address of the server that MQFM is running on
${MQFM_ERROR_TEXT}	The last MQFM error message

10.2 Examples

```
<Copy todir="C:\temp\mqfm" tofile="${FILE}.${DATE}">
  <File dir="data">test*</File>
</Copy>
```

```
<Move todir="C:\temp\mqfm" tofile="${FILE_PREFIX}.${DATE}.${FILE_EXT}">
  <File dir="data">test*</File>
</Move>
```

```
<Send delete="N" format="N">
  <File>data\test.xml</File>
  <MQ>
    <QMgrName>MQWT1</QMgrName>
    <QueueName>TEST.Q1</QueueName>
  </MQ>
  <Remote>
    <Directory>C:\temp</Directory>
    <FileName>${FILE}.${DATE}</FileName>
    <Execute xmlfile="abcrun.xml" />
  </Remote>
</Send>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE MQFM_Job SYSTEM "MQFM_Job.dtd">
<MQFM_Job>
  <Job name="alert_pgm">
    <Command wait='y'>C:\test\alert.exe</Command>
    <Parm>-m</Parm>
    <Parm>${HOSTNAME} - ${MQFM_ERROR_TEXT}</Parm>
  </Job>
</MQFM_Job>
```

11 Appendix A – Custom Logging

MQFM supports user-defined custom logger via Apache's log4j framework.

Apache log4j has an abstract class, org.apache.log4j.AppenderSkeleton, that implements most of the functionality needed for writing a custom Appender. When AppenderSkeleton class is extended, 3 methods need to be implemented: *append*, *requiresLayout* and *close*.

A sample biz.capitalware.mqfm.logging.CustomAppend is included in the source as a working sample:

```
package biz.capitalware.mqfm.logging;

import org.apache.log4j.AppenderSkeleton;
import org.apache.log4j.spi.LoggingEvent;

/**
 * Sample custom appender for MQFM.
 * @author Roger Lacroix, Capitalware Inc.
 * @version 1.0
 * @license Apache 2 License
 */
public class CustomAppender extends AppenderSkeleton
{
   protected void append(LoggingEvent loggingEvent)
   {
      System.out.println("CustomAppender: "+loggingEvent.getMessage());
   }

   public boolean requiresLayout()
   {
      return false;
   }

   public void close()
   {
   }
}
```

- ***void append(LoggingEvent)***: This method will be called by log4j at runtime when an output has to occur. We can extract the information from the LoggingEvent class and perform some appropriate function. Later on, we will see how to extract information from the LoggingEvent class and send a Yahoo instant message.
- ***boolean requiresLayout()***: This method is used to indicate whether the custom appender requires a Layout. Since we are going to keep things simple, we will simply return false.
- ***void close()***: This method is called by log4j runtime as a signal to release any resources such as file handles, network resources, etc. With no resources to release, we can simply keep it empty.

In addition to the above methods, one needs to write a get/set method pair for each custom property that will be configured for the appender. The setting of the properties will be done transparently at runtime by log4j.

Next, compile and your CustomAppender to the CLASSPATH of batch file or Unix shell script. If you are compiling the entire MQFM source code then simply replace MQFM's CustomAppender with your CustomAppender.

Finally, to use CustomAppender, the log4j.properties file will need to be updated:

```
#
# MQFM
#
log4j.category.MQFM=INFO, mqfm, stdout, custom
#
# "mqfm" appender writes to a file
#
log4j.appender.mqfm=org.apache.log4j.RollingFileAppender
log4j.appender.mqfm.File=log/MQFM.log
log4j.appender.mqfm.MaxFileSize=1000KB
log4j.appender.mqfm.MaxBackupIndex=9
log4j.appender.mqfm.layout=org.apache.log4j.PatternLayout
log4j.appender.mqfm.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
#
# CustomAppender
#
log4j.appender.custom=biz.capitalware.mqfm.logging.CustomAppender
log4j.appender.custom.layout=org.apache.log4j.PatternLayout
log4j.appender.custom.layout.ConversionPattern=%m%n
```

12 Appendix B – MQ File Mover Upgrade Procedures

To upgrade an existing installation of MQ File Mover, please do the following in the appropriate section below.

12.1 Windows Upgrade

- Stop all instances of Watch and Receive
- Backup all MQFM data files in the MQFM install directory
- Delete the MQFM install directory
- Unzip mqfm.zip archive
- Restore the MQFM data files if necessary

12.2 Unix and Linux Upgrade

- Stop all instances of Watch and Receive
- Backup all MQFM data files in the MQFM install directory
- Delete the MQFM install directory
- Unzip mqfm.zip archive
- Restore the MQFM data files if necessary

12.3 IBM i Upgrade

- Stop all instances of Watch and Receive
- Backup all MQFM data files in the MQFM install directory
- Delete the MQFM install directory
- Unzip mqfm.zip archive
- Restore the MQFM data files if necessary

13 Appendix C – Support

The support for MQ File Mover can be found at the following location (requires a support contract):

Online Help Desk Ticketing System at
www.capitalware.biz/phpst/

By email at:
support@capitalware.biz

By regular mail at:

Capitalware Inc.
Attn: MQ File Mover Support
1673 Richmond Street, Suite 524
London, Ontario N6G2N3
Canada

14 Appendix D – Summary of Changes

- MQ File Mover v4.1.8
  - Fixed an issue with the Action's checkObject not setting the error text in the GlobalErrorText.

- MQ File Mover v4.1.7
  - Fixed an issue with the Receive action when the same file is received again but is smaller than the original.

- MQ File Mover v4.1.6
  - Fixed an issue with the Receive action using run='S' and putting an empty message to the backout queue.

- MQ File Mover v4.1.5
  - Fixed an issue with the CCDTFile element

- MQ File Mover v4.1.4
  - Fixed a token issue in the SendEmail Action
  - Fixed a path issue in the Zip Action

- MQ File Mover v4.1.3
  - Fixed a bug in the Receive Action's Default -> FileName checking for a null and/or blank

- MQ File Mover v4.1.2
  - Added usefilelocking attribute for Global element for MQFM Workflow
  - Fixed CLASSPATH for mqfm.sh and mqfm64.sh

- MQ File Mover v4.1.1
  - Fixed a bug with Archiving files when the file is greater than 4190000 bytes
  - Fixed a bug with exception handling for Receive Action when directory does not exist

- MQ File Mover v4.1.0
  - Updated Zip Action to handle locked files (locked by another process)
  - Updated Zip Action to handle Windows drive label (i.e. C:)
  - Updated Watch Action to perform retry on a failure
  - Added the ability to use Global properties with other Global properties
  - Added code to dump out user global values
  - Added createdir attribute for Append, Copy, DecryptFile, EncryptFile, Merge, MergeSort, Move, Receive, ReplaceText, Sort, Tar, Touch, UnTar, UnZip, & Zip Actions
  - Added append attribute for Receive Action
  - Added default value ("*") for Extension Element for Watch Action for type="D"

  - Added createdir attribute for Archive Element
  - Fixed bug in XML processing for Global variables
  - Fixed bug in XML processing for Execute Object

- MQ File Mover v4.0.0
  - Added support for unlimited file size for Send, Receive and Watch Actions
  - Added support for sending the file to a single queue or to multiple queues for Send and Watch Actions
  - Added support for end-to-end encryption:
    - Enhanced Send and Watch Actions to have the ability to encrypt a message with AES 128, 192 or 256-bit.
    - Enhanced Receive Action to have the ability to decrypt a message with AES 128, 192 or 256-bit.
  - Added support for file locking when reading and writing to/from a file.
  - All Actions now have an “On Error Fail” flag.
  - Added a DecryptFile Action to decrypt a file with AES 128, 192 or 256-bit.
  - Added a EncryptFile Action to encrypt a file with AES 128, 192 or 256-bit.
  - Added an If/Else Action to perform a conditional test against an Action's variable
  - Added a Launch Action to invoke an MQFM Workflow XML file.
  - Added a Merge Action to merge 2 or more files into a target file
  - Added a MergeSort Action to merge 2 or more files and sort the results into a target file
  - Added a ReplaceText Action to perform a search and replace of text on a file.
  - Added a Schedule Action to invoke an MQFM Workflow XML file at a specific date and/or time.
  - Added a SendEmail Action to send an email to 1 or more recipients.
  - Added a Sleep Action to pause the Workflow.
  - Added a Sort Action to sort a file into a target file
  - Added a Touch Action to set the modification time of the file to the current time of day. If the file doesn't exist, it is created.
  - Added a Tar Action to create a tar archive from the contents of a file or files in a directory.
  - Added an UnTar Action to extract the contents of a tar archive to a directory.
  - Added an UnZip Action to extract the contents of a zip archive to a directory.
  - Added support for custom logging.
  - Fixed an issue with "dir" attribute of File element of Watch element

- MQ File Mover v3.2.1.1
  - Updated the MQFM.sh and MQFM64.sh shell scripts to support IBM i (OS/400)

- MQ File Mover v3.2.1
  - Fixed an issue with Watch Action and token processing with Archive File name.

- MQ File Mover v3.2.0
  - Fixed an issue with MQFM running as a Windows service when WMQ v7.0 is installed locally

  - Fixed an issue with exception handling when MQFM is running as a Windows service

- MQ File Mover v3.1.0
  - Added WatchSend tag under the Watch tag in the MQFM_Watch XML DTD
  - Fixed a bug in the Receive action's Execute tag
  - Fixed token processing for copy, move, rename and zip

- MQ File Mover v3.0.0
  - Major rewrite
  - Added support for an Action Framework
  - Added the ability to monitor for a particular file or monitor a directory for files to appear
  - Added an Execute Action to run an external program / application
  - Added file actions: Append, Copy, Move, Delete, Rename and Zip
  - Added token replacement for target file name for actions: Copy, Execute, Move, Rename, Receive, Send and Zip
  - Added support running MQFM as a Windows Service.

- MQ File Mover v2.3.0
  - Added support for archiving the file in File2Msg
  - Added support for deleting the file in File2Msg
  - Created a mechanism (JobName) for specifying the command, parameters and actions (wait, archive or delete) to launch an external program.
  - Added support for launching a local JobName in File2Msg
  - Added support for specifying a remote JobName in File2Msg
  - Added support for launching a JobName in Msg2File

- MQ File Mover v2.2.0
  - Added support for wildcards in the filename for File2Msg

- MQ File Mover v2.1.0
  - Added support for on-the-fly compression and decompression of the message data
  - Added support for CCDT (Client Channel Definition Tables) to connect to a remote queue manager.
  - Added code to explicitly set the message to be persistent in File2Msg.

- MQ File Mover v2.0.0
  - Major rewrite to support 'MQ File Mover' messages.

- MQ File Mover v1.0.0
  - Initial release.

15 Appendix E – License Agreement

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

16 Appendix F – Notices

Trademarks:

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

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

Intel is a registered trademark of Intel Corporation.

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

Linux is a trademark of Linus Torvalds.

Mac OS X is a trademark of Apple Computer Inc.

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

UNIX is a registered trademark of the Open Group.

WebLogic is a trademark of BEA Systems Inc.