

MQ Channel Encryption for z/OS Installation and Operation Manual



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

Last Updated: July 2020.
© Copyright Capitalware Inc. 2010, 2020.

Table of Contents

1 INTRODUCTION.....	1
1.1 OVERVIEW.....	1
1.2 EXECUTIVE SUMMARY.....	2
1.3 MESSAGE DIAGRAM (LOGICAL VIEW).....	2
1.4 CONTEXT DIAGRAM (LOGICAL VIEW).....	3
1.5 PREREQUISITES.....	4
1.5.1 <i>Operating System</i>	4
1.5.2 <i>IBM MQ</i>	4
2 INSTALLING MQ CHANNEL ENCRYPTION FOR Z/OS.....	5
2.1 CHANNEL EXIT.....	5
2.1.1 <i>z/OS Installation</i>	5
2.1.2 <i>z/MQCE Datasets</i>	6
2.1.3 <i>z/OS CHIN JCL</i>	7
3 CONFIGURING QMGR TO QMGR CHANNELS.....	8
3.1 MESSAGE EXIT DATA (MSGDATA).....	9
3.1.1 <i>MSGDATA with DD Name</i>	9
3.1.2 <i>MSGDATA with DD Name and Member Name</i>	10
3.2 SENDER CHANNEL.....	11
3.3 RECEIVER CHANNEL.....	11
3.4 SERVER CHANNEL.....	12
3.5 REQUESTER CHANNEL.....	12
3.6 CLUSTER SENDER CHANNEL.....	13
3.7 CLUSTER RECEIVER CHANNEL.....	13
4 CONFIGURING CLIENT CHANNELS.....	14
4.1 USER DATA (SENDDATA AND RCVDATA).....	15
4.1.1 <i>User Data with DD Name</i>	15
4.1.2 <i>User Data with DD Name and Member Name</i>	16
4.2 SERVER CONNECTION CHANNEL.....	17
5 INIFILE KEYWORDS.....	18
5.1 LOGGING.....	18
5.2 KEYSIZE.....	18
5.3 PERFORM.....	18
5.4 ENCPASSPHRASE, PASSPHRASE AND USEPP.....	19
5.5 CONVERTPP.....	20
5.6 ALLSEGMENTS.....	20
5.7 LICENSEFILE.....	21
5.8 LICENSE KEY.....	21
6 APPENDIX A – Z/MQCE INIFILE.....	22
7 APPENDIX B – Z/MQCE UPGRADE PROCEDURES.....	25
8 APPENDIX C - ENCRYPT PASSPHRASE.....	27

8.1	EXAMPLES.....	27
8.1.1	<i>z/OS</i>	27
9	APPENDIX D – CAPITALWARE PRODUCT DISPLAY VERSION.....	28
9.1	EXAMPLES.....	28
9.1.1	<i>z/OS</i>	28
10	APPENDIX E - EXPLICITLY SETTING VALUES IN MSGDATA, RCVDATA & SENDDATA.....	29
11	APPENDIX F – ENCRYPTION AND DIGITAL SIGNATURE.....	30
12	APPENDIX G – SUPPORT.....	31
13	APPENDIX H – SUMMARY OF CHANGES.....	32
14	APPENDIX I – LICENSE AGREEMENT.....	34
15	APPENDIX J – NOTICES.....	36

1 Introduction

1.1 Overview

MQ Channel Encryption for z/OS (z/MQCE) provides encryption for MQ message data. In cryptography, encryption is the process of transforming information into an unreadable form (encrypted data). Decryption is the reverse process. It makes the encrypted information readable again. Only those with the key (PassPhrase) can successfully decrypt the encrypted data.

z/MQCE provides encryption for message data, which flows between IBM MQ (MQ) resources. z/MQCE operates with MQ v5.3.1, v6.0, v7.0, v6.0, v7.0, v7.1, v8.0, v9.0, v9.1 and v9.2 for z/OS environments. It operates with Sender, Receiver, Server, Requester, Cluster-Sender, Cluster-Receiver, Server Connection and Client Connection channels of the MQ queue managers.

z/MQCE is a simple drop-in solution that provides cryptographic protection for MQ queue managers. The protection can be queue manager to queue manager or client application to queue manager.

- Queue manager to queue manager protection means all messages flowing over a channel between 2 queue managers will be encrypted.
- Client application to queue manager protection means application-level message data flowing between a MQ client application and queue manager will be encrypted.

The z/MQCE can be configured as a queue manager channel message exit or as a channel sender/receive exit pair.

z/MQCE uses Advanced Encryption Standard (AES) to encrypt the data. AES is a data encryption scheme, adopted by the US government, that uses three different key sizes (128-bit, 192-bit, and 256-bit). AES was announced by National Institute of Standards and Technology (NIST) as U.S. FIPS PUB 197 (FIPS 197) on November 26, 2001 after a 5-year standardization process.

z/MQCE uses the SHA-2 to create a cryptographic hash function (digital signature) for the message data.

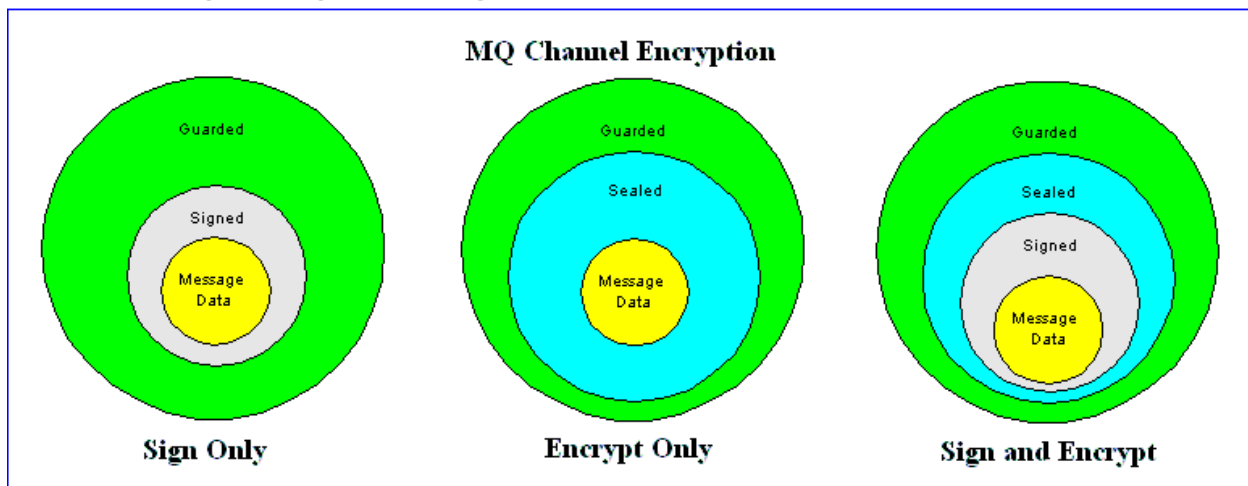
1.2 Executive Summary

The z/MQCE solution is an MQ encryption exit. It is available for z/OS v1.4 or higher environments.

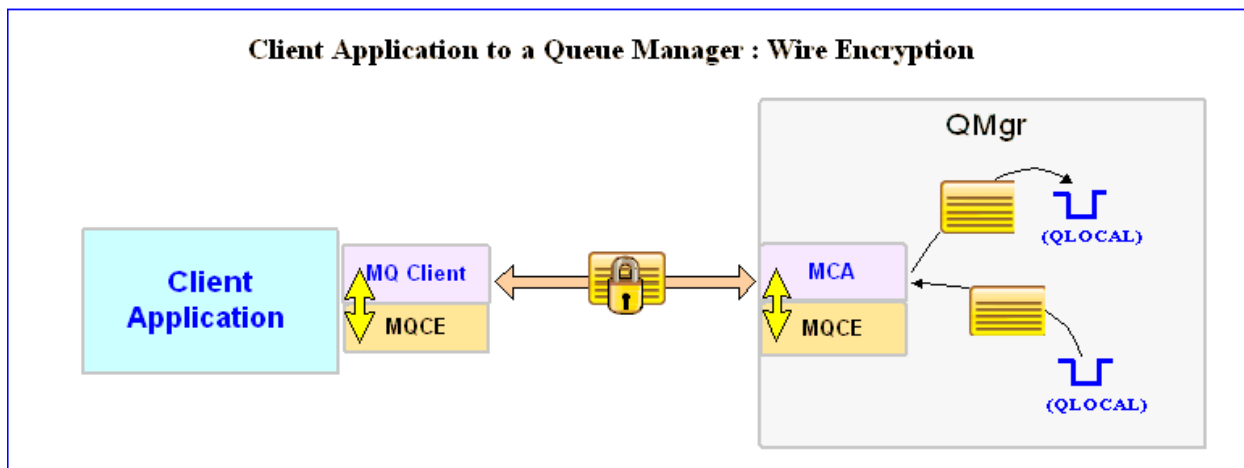
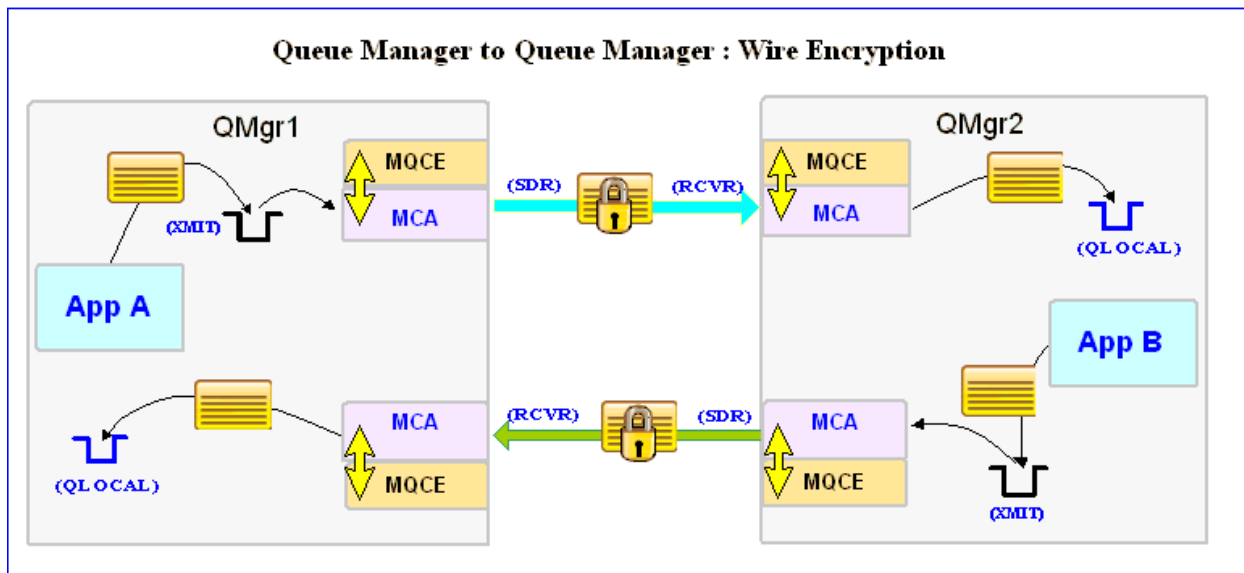
Major Features of MQCE for z/OS:

- Easy to set up and configure (unlike SSL)
- No application changes required
- Can be configured as either queue manager to queue manager or client application to queue manager solution
- For both modes, all message data flowing over a channel will be encrypted (nothing missed or forgotten)
- Secure encryption/decryption methodology using AES with 128, 192 or 256-bit keys
- Uses the SHA-2 to create a cryptographic hash function (digital signature)
- Standard MQ feature, GET-with-Convert, is supported
- Provides logging capability via Write To Operator (WTO) facility.

1.3 Message Diagram (Logical View)



1.4 Context Diagram (Logical View)



1.5 Prerequisites

This section details the minimum supported software levels. These prerequisites apply to both client-side and server-side installations of MQ Channel Encryption for z/OS.

1.5.1 Operating System

MQ Channel Encryption for z/OS can be installed on any of the following supported servers:

1.5.1.1 IBM z/OS

- IBM z/OS v1.4 or higher

1.5.2 IBM MQ

- IBM MQ for z/OS v5.3.1, v6.0, v7.0, v6.0, v7.0, v7.1, v8.0, v9.0, v9.1 and v9.2

2 Installing MQ Channel Encryption for z/OS

This section describes how to install Capitalware's MQ Channel Encryption for z/OS.

2.1 Channel Exit

The following files are the platform specific channel exits and the required initialization file (IniFile).

2.1.1 z/OS Installation

To install the MQCE for z/OS, first unzip the **mqce_zos-setup.zip**. The zip file contains 2 z/OS XMIT prepared datasets.

- **MQCE.LOAD.ZOS** is the XMIT dataset that contains the z/OS load-module.
- **MQCE.SYSIN.ZOS** is the XMIT dataset that contains a sample initialization file for the channel exit and sample MQSC script to define MQ channels with the channel exit.

Steps to install the channel exit:

1. ftp the z/OS XMIT prepared datasets to the z/OS LPAR.

```
ftp -s:mqce.ftp z/OS_hostname
```

```
your-z/OS-userid  
your-z/OS-password  
  
binary  
quote SITE recfm=fb lrecl=80 blksize=3120  
put MQCE.LOAD.ZOS  
put MQCE.SYSIN.ZOS  
quit
```

If the user receives the following error message then they will need to pre-allocate the z/OS datasets:

```
ftp> put MQCE.LOAD.ZOS  
200 Port request OK.  
550-SVC99 RETURN CODE=4 S99INFO=0 S99ERROR=38656 HEX=9700 S99ERSN code X'000003F3'.  
550 Unable to create data set xxxxx.MQCE.LOAD.ZOS for STOR command.  
ftp> put MQCE.SYSIN.ZOS  
200 Port request OK.  
550-SVC99 RETURN CODE=4 S99INFO=0 S99ERROR=38656 HEX=9700 S99ERSN code X'000003F3'.  
550 Unable to create data set xxxxx.MQCE.SYSIN.ZOS for STOR command.
```

To pre-allocating the XMIT datasets go to option 3.2 of ISPF and allocate both datasets: MQCE.LOAD.ZOS and MQCE.SYSIN.ZOS.

Use the following dataset attributes when allocating both datasets:

Space	
Units	BLOCKS
Primary Quantity	40
Secondary Quantity	40
Directory Blocks	0
DCB Parameters	
RECFM	FB
LRECL	80
BLKSIZE	3120
DsnType	Blank

After the user has pre-allocated the datasets, they can redo the ftp commands.

2. Log on to z/OS LPAR and issue the following TSO RECEIVE commands:

```
TSO RECEIVE INDATASET(MQCE.LOAD.ZOS)
TSO RECEIVE INDATASET(MQCE.SYSIN.ZOS)
```

After issuing the above commands, the following product datasets will appear:

- **+HLQ+.CPTLWARE.MQCE.LOAD** is the dataset that contains the z/OS load-module.
- **+HLQ+.CPTLWARE.MQCE.SYSIN** is a dataset that contains a sample initialization file for the channel exit and sample MQSC script to define MQ channels with the channel exits.

2.1.2 z/MQCE Datasets

z/MQCE solution is comprised of 2 datasets: +HLQ+.CPTLWARE.MQCE.LOAD and +HLQ+.CPTLWARE.MQCE.SYSIN.

2.1.2.1 +HLQ+.CPTLWARE.MQCE.LOAD

- **MQCE** is the actual channel exit z/OS load-module that will be invoked by the MQ Server component.

2.1.2.2 +HLQ+.CPTLWARE.MQCE.SYSIN

- **MQCEINI** is a sample initialization file for the channel exit.
- **SSXMQSC** is a sample MQSC script to define MQ channels with the channel exit.

2.1.3 z/OS CHIN JCL

This section describes the required JCL for z/MQCE.

2.1.3.1 CSQXLIB DDName

The MQCE load-module needs to be put in the executable path for the CHINIT started-task. There are 2 options for achieving this:

1. Add the dataset to the CSQXLIB concatenation of the CHINIT's CSQXLIB.

```
//CSQXLIB DD DISP=SHR,DSN=+MQHLQ+. +QMGRNAME+. USERAUTH  
// DD DISP=SHR,DSN=+HLQ+. CPTLWARE.MQCE.LOAD
```

2. Or copy the MQCE load-module to your existing MQ exit / link-edited parameter dataset. Here is a sample JCL to copy the MQCE load-module:

```
//COPY1 EXEC PGM=IEBCOPY,REGION=1024K  
//SYSPRINT DD SYSOUT=*  
//SYSUT3 DD DSN=&&SYSUT3,UNIT=SYSDA,DISP=(,DELETE),  
// SPACE=(CYL,(5,1))  
//SYSUT4 DD DSN=&&SYSUT4,UNIT=SYSDA,DISP=(,DELETE),  
// SPACE=(CYL,(5,1))  
//*  
//IN DD DISP=SHR,DSN=+HLQ+. CPTLWARE.MQCE.LOAD  
//*  
//OUT DD DISP=SHR,DSN=+MQHLQ+. +QMGRNAME+. USERAUTH  
//*  
//SYSIN DD *  
COPYMOD OUTDD=OUT,INDD=((IN,R))  
S M=MQCE  
/*
```

2.1.3.2 MQCEIN DDName

MQCEIN is the DDName that points to a dataset containing the IniFile parameters.

Add the following line to the CHINIT's JCL.

```
//MQCEIN DD DISP=SHR,DSN=+HLQ+. CPTLWARE.MQCE.SYSIN(MQCEINI)
```

3 Configuring QMgr to QMgr Channels

This section describes how to configure the encryption exit.

For normal operation of the z/MQCE solution, all that is required is the license key, supplied by Capitalware Inc. in the MSGDATA attribute field.

- MSGDATA
L=0000-AAAA-BBBBBBBB

Where '0000-AAAA-BBBBBBBB' is the license key supplied by Capitalware Inc.

For debugging purpose, the MQ Admin can specify an IniFile in the MSGDATA attribute.

- For z/OS, the MSGDATA attribute would be:
MQCEIN

Note: Message Exit Data must NOT exceed 32 characters.

3.1 Message Exit Data (MSGDATA)

MQCE supports 2 ways to specify an IniFile via the Message Exit Data (MSGDATA) field: DD Name and DD Name with a Member Name.

3.1.1 MSGDATA with DD Name

In this case, only the DD Name is used to specify the IniFile. The DD Name provided in the MSGDATA field must match the DD Name in the CHIN's JCL. The DD statement's DSN keyword can contain either a fully qualified Partition DataSet with the Member name or a Sequential DataSet.

3.1.1.1 MSGDATA with DD Name using Partition DataSet

The CHIN's DD Name references the DSN keyword which contains the fully qualified Partition DataSet Name (highlighted in **red**) and member name (highlighted in **blue**). Since the Member Name is included in the CHIN'S DD DSN keyword, do not put the Member Name in the MSGDATA field.

e.g.

```
MSGDATA('DDName')
```

CHIN JCL using Partition DataSet

```
//MQCEIN DD DISP=SHR,DSN=+HLQ+.CPTLWARE.MQCE.SYSIN(MQCEINI)
```

```
DEFINE CHANNEL ('MQA1.TO.MQB1') CHLTYPE(SDR) +
  TRPTYPE(TCP) +
  XMITQ('MQB1.XMIT') +
  CONNAME('127.0.0.1(1415)') +
  MSGEXIT('MQCE') +
  MSGDATA('MQCEIN') +
  REPLACE
```

3.1.1.2 MSGDATA with DD Name using Sequential DataSet

The CHIN's DD Name specifies a DSN which will contain the Sequential DataSet. As seen below, the DD Name in the MSGDATA field matches the DD Name in the CHIN's JCL.

e.g.

```
MSGDATA('DDName')
```

CHIN JCL using Sequential DataSet

```
//MQCEIN DD DISP=SHR,DSN=+HLQ+.CPTLWARE.MQCE.SYSIN.SEQ
```

```
DEFINE CHANNEL ('MQA1.TO.MQB1') CHLTYPE(SDR) +
  TRPTYPE(TCP) +
  XMITQ('MQB1.XMIT') +
  CONNAME('127.0.0.1(1415)') +
  MSGEXIT('MQCE') +
  MSGDATA('MQCEIN') +
  REPLACE
```

3.1.2 MSGDATA with DD Name and Member Name

In this case, both the DD Name (highlighted in **red**) and the Member Name (highlighted in **blue**) are used to specify the IniFile since the DSN keyword of the DD statement only contains the Partition DataSet Name. In other words, the user specifies the Member Name as a parameter to the MSGDATA field. This is a dynamic configuration that allows for different IniFiles for different channels.

e.g.

```
MSGDATA('DDName(MemberName)')
```

CHIN JCL using Partition DataSet

```
//MQCEIN DD DISP=SHR,DSN=+HLQ+.CPTLWARE.MQCE.SYSIN
```

```
DEFINE CHANNEL ('MQA1.TO.MQB1') CHLTYPE(SDR) +  
  TRPTYPE(TCP) +  
  XMITQ('MQB1.XMIT') +  
  CONNAME('127.0.0.1(1415)') +  
  MSGEXIT('MQCE') +  
  MSGDATA('MQCEIN(MQCEINI)') +  
  REPLACE
```

3.2 Sender Channel

This section describes the necessary entries to enable the encryption exit. The MQ Administrator will need to update 2 fields of the SENDER Channel that the encryption exit will be applied to.

Note: The Message Exit Data (MSGDATA) field must NOT exceed 32 characters.

On z/OS, MSGEXIT and MSGDATA will contain the following values assuming a default install.

- MSGEXIT
MQCE

```
DEFINE CHANNEL ('MQA1.TO.MQB1') CHLTYPE(SDR) +
  TRPTYPE(TCP) +
  XMITQ('MQB1.XMIT') +
  CONNAME('127.0.0.1(1415)') +
  MSGEXIT('MQCE') +
  MSGDATA('L=0000-AAAA-BBBBBBBB') +
  REPLACE
```

3.3 Receiver Channel

This section describes the necessary entries to enable the encryption exit. The MQ Administrator will need to update 2 fields of the RECEIVER Channel that the encryption exit will be applied to.

Note: The Message Exit Data (MSGDATA) field must NOT exceed 32 characters.

On z/OS, MSGEXIT and MSGDATA will contain the following values assuming a default install.

- MSGEXIT
MQCE

```
DEFINE CHANNEL('MQB1.TO.MQA1') CHLTYPE(RCVR) +
  TRPTYPE(TCP) +
  MSGEXIT('MQCE') +
  MSGDATA('L=0000-AAAA-BBBBBBBB') +
  REPLACE
```

3.4 Server Channel

This section describes the necessary entries to enable the encryption exit. The MQ Administrator will need to update 2 fields of the SERVER Channel that the encryption exit will be applied to.

Note: The Message Exit Data (MSGDATA) field must NOT exceed 32 characters.

On z/OS, MSGEXIT and MSGDATA will contain the following values assuming a default install.

- MSGEXIT
MQCE

```
DEFINE CHANNEL ('MQA1.TO.MQB1') CHLTYPE(SVR) +
  TRPTYPE(TCP) +
  XMITQ('MQB1.XMIT') +
  CONNAME('127.0.0.1(1415)') +
  MSGEXIT('MQCE') +
  MSGDATA('L=0000-AAAA-BBBBBBBB') +
  REPLACE
```

3.5 Requester Channel

This section describes the necessary entries to enable the encryption exit. The MQ Administrator will need to update 2 fields of the REQUESTOR Channel that the encryption exit will be applied to.

Note: The Message Exit Data (MSGDATA) field must NOT exceed 32 characters.

On z/OS, MSGEXIT and MSGDATA will contain the following values assuming a default install.

- MSGEXIT
MQCE

```
DEFINE CHANNEL('MQB1.TO.MQA1') CHLTYPE(RQSTR) +
  TRPTYPE(TCP) +
  MSGEXIT('MQCE') +
  MSGDATA('L=0000-AAAA-BBBBBBBB') +
  REPLACE
```


3.6 Cluster Sender Channel

This section describes the necessary entries to enable the encryption exit. The MQ Administrator will need to update 2 fields of the CLUSSDR Channel that the encryption exit will be applied to.

Note: The Message Exit Data (MSGDATA) field must NOT exceed 32 characters.

On z/OS, MSGEXIT and MSGDATA will contain the following values assuming a default install.

- MSGEXIT
MQCE

```
DEFINE CHANNEL ('MQA1.TO.MQB1') CHLTYPE(CLUSSDR) +  
  TRPTYPE(TCP) +  
  XMITQ('MQB1.XMIT') +  
  CONNAME('127.0.0.1(1415)') +  
  MSGEXIT('MQCE') +  
  MSGDATA('L=0000-AAAA-BBBBBBBB') +  
  REPLACE
```

3.7 Cluster Receiver Channel

This section describes the necessary entries to enable the encryption exit. The MQ Administrator will need to update 2 fields of the CLUSRCVR Channel that the encryption exit will be applied to.

Note: The Message Exit Data (MSGDATA) field must NOT exceed 32 characters.

On z/OS, MSGEXIT and MSGDATA will contain the following values assuming a default install.

- MSGEXIT
MQCE

```
DEFINE CHANNEL ('MQB1.TO.MQA1') CHLTYPE(CLUSRCVR) +  
  TRPTYPE(TCP) +  
  MSGEXIT('MQCE') +  
  MSGDATA('L=0000-AAAA-BBBBBBBB') +  
  REPLACE
```

4 Configuring Client Channels

This section describes how to configure the encryption exit.

For normal operation of the z/MQCE solution, all that is required is the license key, supported by Capitalware Inc., in the MSGDATA attribute field.

- MSGDATA
L=0000-AAAA-BBBBBBBB

For debugging purpose, the MQ Admin can specify an IniFile in the MSGDATA attribute.

- For z/OS, the MSGDATA attribute would be:
MQCEIN

Note: The Send / Receive Exit Data field must NOT exceed 32 characters.

4.1 User Data (SENDDATA and RCVDATA)

MQCE supports 2 ways to specify an IniFile via the User Data (SENDDATA and RCVDATA) field: DD Name and DD Name with a Member Name.

4.1.1 User Data with DD Name

In this case, only the DD Name is used to specify the IniFile. The DD Name provided in the MSGDATA field must match the DD Name in the CHIN's JCL. The DD statement's DSN keyword can contain either a fully qualified Partition DataSet with the Member name or a Sequential DataSet.

4.1.1.1 User Data with DD Name using Partition DataSet

The CHIN's DD Name references the DSN keyword which contains the fully qualified Partition DataSet Name (highlighted in **red**) and member name (highlighted in **blue**). Since the Member Name is included in the CHIN'S DD DSN keyword, do not put the Member Name in the SENDDATA or RCVDATA fields.

e.g.

```
SENDDATA('DDName')
```

CHIN JCL using Partition DataSet

```
//MQCEIN DD DISP=SHR,DSN=+HLQ+.CPTLWARE.MQCE.SYSIN(MQCEINI)
```

```
DEFINE CHANNEL ('MQA1.APP.CH01') CHLTYPE(SVRCONN) +  
  TRPTYPE(TCP) +  
  SENDEXIT('MQCE') +  
  SENDDATA('MQCEIN') +  
  RCVEXIT('MQCE') +  
  RCVDATA('MQCEIN') +  
  REPLACE
```

4.1.1.2 User Data with DD Name using Sequential DataSet

The CHIN's DD Name specifies a DSN which will contain the Sequential DataSet. As seen below, the DD Name in the SENDDATA or RCVDATA fields match the DD Name in the CHIN's JCL.

e.g.

```
SENDDATA('DDName')
```

CHIN JCL using Sequential DataSet

```
//MQCEIN DD DISP=SHR,DSN=+HLQ+.CPTLWARE.MQCE.SYSIN.SEQ
```

```
DEFINE CHANNEL ('MQA1.APP.CH01') CHLTYPE(SVRCONN) +  
  TRPTYPE(TCP) +  
  SENDEXIT('MQCE') +  
  SENDDATA('MQCEIN') +  
  RCVEXIT('MQCE') +  
  RCVDATA('MQCEIN') +  
  REPLACE
```

4.1.2 User Data with DD Name and Member Name

In this case, both the DD Name (highlighted in **red**) and the Member Name (highlighted in **blue**) are used to specify the IniFile since the DSN keyword of the DD statement only contains the Partition DataSet Name. In other words, the user specifies the Member Name as a parameter to the MSGDATA field. This is a dynamic configuration that allows for different IniFiles for different channels.

e.g.

```
SENDDATA('DDName(MemberName)')
```

CHIN JCL using Partition DataSet

```
//MQCEIN DD DISP=SHR,DSN=+HLQ+.CPTLWARE.MQCE.SYSIN
```

```
DEFINE CHANNEL ('MQA1.APP.CH01') CHLTYPE(SVRCONN) +  
  TRPTYPE(TCP) +  
  SENDEXIT('MQCE') +  
  SENDDATA('MQCEIN(MQCEINI)') +  
  RCVEXIT('MQCE') +  
  RCVDATA('MQCEIN(MQCEINI)') +  
  REPLACE
```

4.2 Server Connection Channel

This section describes the necessary entries to enable the server-side encryption exit. The MQ Administrator will need to update 2 fields of the SVRCONN Channel that the server-side encryption exit will be applied to.

Note: The Send / Receive Exit Data field must NOT exceed 32 characters.

On z/OS, SENDEXIT, SENDDATA, RCVEXIT and RCVDATA will contain the following values assuming a default install.

- SENDEXIT
MQCE
- RCVEXIT
MQCE

```
DEFINE CHANNEL ('MQA1.APP.CH01') CHLTYPE(SVRCONN) +  
  TRPTYPE(TCP) +  
  SENDEXIT('MQCE') +  
  SENDDATA('L=0000-AAAA-BBBBBBBB') +  
  RCVEXIT('MQCE') +  
  RCVDATA('L=0000-AAAA-BBBBBBBB') +  
  REPLACE
```

5 IniFile Keywords

5.1 Logging

This section describes the necessary entries to enable z/MQCE to write log information. To enable and control logging, you need 3 keywords in the IniFile:

- **LogMode** specifies what type of logging the user wishes to have. LogMode supports 4 values [Q / N / V / D] where Q is Quiet, N is Normal, V is Verbose and D is Debug. The default value is Q.
- **LogFile** LogFile specifies the location of the log file. The default is as follows:

LogFile=SYSPRINT

- **UseFormFeed** specifies that a FormFeed command be issued once a day at midnight. UseFormFeed supports 2 values [Y / N]. The default value is N.

```
LogMode=Q
LogFile=SYSPRINT
```

5.2 KeySize

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

```
KeySize=128
```

5.3 Perform

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

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

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

```
Perform=E
```

5.4 EncPassPhrase, PassPhrase and UsePP

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

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

What not to use for your PassPhrase:

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

What to use for your PassPhrase:

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

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

```
UsePP=Y  
EncPassPhrase=jXzFNlKKwZ52wsQ3CUwqWUBpDaovRDnLMDkNqhVEOcswMA
```

Plain text PassPhrase:

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

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

5.5 ConvertPP

This section describes the necessary steps to enable the converting of the PassPhrase from EBCDIC to ASCII. If this IniFile parameter is set to Yes then the specified PassPhrase will be converted from EBCDIC to ASCII.

- **ConvertPP** allows the PassPhrase to be converted from EBCDIC to ASCII

```
ConvertPP=Y
```

5.6 AllSegments

AllSegments specifies whether or not all TSH segments will be encrypted. AllSegments supports 2 values [Y / N]. The default value is N.

- **Y** means that z/MQCE will encrypt all TSH segments that are sent
- **N** means that z/MQCE will only encrypt the TSH segments that contain message data

```
AllSegments=N
```


5.7 LicenseFile

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

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

QMgrName = License_Key

Example:

```
MQA1 = 10C0-AAAA-BBBBBBBB  
MQB1 = 10C0-XXXX-CCCCCCCC
```

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

The following are the default values for LicenseFile:

For z/OS DD:

LicenseFile=MQCEFILE

5.8 License Key

This section will describe how to license MQ Channel Encryption for z/OS to a particular queue manager.

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

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

```
License=10C0-AAAA-BBBBBBBB
```

6 Appendix A – z/MQCE IniFile

The sample IniFile below is the MQCEINI file supplied for z/OS. The IniFile supports the following keywords and their values:

```
LogMode=N
LogFile=SYSPRINT
KeySize=128
License=
```

Note: Keywords are case sensitive.

Keyword	Description of Server-side keywords
AllSegments	<p>AllSegments specifies whether or not all TSH segments will be encrypted. AllSegments set to 'N' means only data segments will be encrypted. AllSegments supports 2 values [Y / N]. The default value is N.</p> <p>e.g. AllSegments=Y</p>
ConvertPP	<p>ConvertPP specifies whether or not to convert the PassPhrase from EBCDIC to ASCII. ConvertPP supports 2 values [Y / N]. The default value is Y.</p> <p>e.g. ConvertPP=Y</p>
EncPassPhrase	<p>EncPassPhrase specifies the encrypted PassPhrase for the message encryption and/or decryption. See Appendix C for details on creating the encrypted PassPhrase.</p> <p>e.g. EncPassPhrase=jXzFNIKKwZ52wsQ3CUwqWUBpDaoVRDnLMDkNqhVEOcsWMA</p> <p>Note: Only used if UsePassPhrase is set to 'Y'.</p>
KeySize	<p>KeySize specifies the AES key size used for the encryption / decryption of the message data. Valid values are 128, 192 or 256. The default value is 128.</p> <p>e.g. KeySize=128</p>
License	<p>License specifies the queue manager's license key. Your license will look something like: 0000-AAAA-BBBBBBBB (Note: This is a sample license only and will NOT work).</p> <p>e.g. License=0000-AAAA-BBBBBBBB</p>

Keyword	Description of Server-side keywords
LicenseFile	<p>LicenseFile specifies the location of License file that contains all of the customer's license keys.</p> <p>The following are the default values for LicenseFile:</p> <p>For z/OS DD: LicenseFile=MQCEFILE</p> <p>e.g. LicenseFile=MQCEFILE</p>
LogFile	<p>LogFile specifies the location of the log file. The default is as follows:</p> <p>For z/OS: LogFile=SYSPRINT</p>
LogMode	<p>LogMode specifies what type of logging the user wishes to have. LogMode supports 4 values [Q / N / V / D] where Q is Quiet, N is Normal, V is Verbose and D is Debug. The default value is Q.</p> <p>e.g. LogMode=Q</p>
PassPhrase	<p>PassPhrase specifies a user supplied PassPhrase. The PassPhrase can be one of three sizes: 16, 24 or 32 characters/digits in length.</p> <p>e.g. PassPhrase=QPriiTJmr4j7aQ2PW</p>
Perform	<p>Perform indicates what functionality that MQCE will perform. Perform supports 3 values [S / E / B]. The default value is E.</p> <ul style="list-style-type: none"> • S means that MQCE will only sign the message • E means that MQCE will only encrypt the message • B means that MQCE will sign and encrypt the message <p>When signing the message, MQCE creates the digital signature using cryptographic hash function of SHA-2.</p> <p>e.g. Perform=E</p>
SequenceNumberFlag	<p>SequenceNumberFlag is a z/OS (OS/390) only flag. It states whether or not there are sequence numbers in columns 72 to 80. SequenceNumberFlag supports 2 values [Y / N]. The default value is N.</p> <p>e.g. SequenceNumberFlag = Y</p>

Keyword	Description of Server-side keywords
UseFormFeed	<p>UseFormFeed specifies that a FormFeed command be issued once a day at midnight. UseFormFeed supports 2 values [Y / N]. The default value is N.</p> <p>e.g. UseFormFeed=Y</p>
UsePP	<p>UsePP allows the user to specify their own PassPhrase. UsePP supports 2 values [Y / N]. The default value is N.</p> <p>e.g. UsePP=Y</p>

7 Appendix B – z/MQCE Upgrade Procedures

To upgrade an existing installation of z/MQCE from an older version to a newer version, do please do the following in the appropriate section below.

1. Stop all of the channels using the z/MQCE exit or stop the queue manager's CHIN (channel initiator).
2. ftp the z/OS XMIT prepared datasets to the z/OS LPAR.

ftp -s:mqce.ftp z/OS_hostname

```
your-z/OS-userid  
your-z/OS-password  
  
binary  
quote SITE recfm=fb lrecl=80 blksize=3120  
put MQCE.LOAD.ZOS  
quit
```

If the user receives the following error message then they will need to pre-allocate the z/OS datasets:

```
ftp> put MQCE.LOAD.ZOS  
200 Port request OK.  
550-SVC99 RETURN CODE=4 S99INFO=0 S99ERROR=38656 HEX=9700 S99ERSN code X'000003F3'.  
550 Unable to create data set xxxxx.MQCE.LOAD.ZOS for STOR command.  
ftp> put MQCE.SYSIN.ZOS  
200 Port request OK.  
550-SVC99 RETURN CODE=4 S99INFO=0 S99ERROR=38656 HEX=9700 S99ERSN code X'000003F3'.  
550 Unable to create data set xxxxx.MQCE.SYSIN.ZOS for STOR command.
```

To pre-allocating the XMIT datasets go to option 3.2 of ISPF and allocate both dataset: MQCE.LOAD.ZOS

Use the following dataset attributes when allocating the dataset:

Space	
Units	BLOCKS
Primary Quantity	40
Secondary Quantity	40
Directory Blocks	0
DCB Parameters	
RECFM	FB
LRECL	80
BLKSIZE	3120
DsnType	Blank

After the user has pre-allocated the dataset, the user can redo the ftp commands.

- Log on to z/OS LPAR and issue the following TSO RECEIVE command:

TSO RECEIVE INDATASET(MQCE.LOAD.ZOS)

After issuing the above command, the following product dataset will appear:

+HLQ+.CPTLWARE.MQCE.LOAD is the dataset that contains the z/OS load-module.

- Start all of the channels using the z/MQCE channel exit or restart the queue manager's CHIN.

8 Appendix C - Encrypt PassPhrase

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

Syntax:

enc_pp plain_text_PassPhrase

Where :

1. plain_text_PassPhrase is the user's PassPhrase to be encrypted

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

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

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

Encrypted PassPhrase: jXzFNIKKwZ52wsQ3CUwqWUBpDaoVRDnLMDkNqhVEOcsWMA

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

e.g.

EncPassPhrase=jXzFNIKKwZ52wsQ3CUwqWUBpDaoVRDnLMDkNqhVEOcsWMA

8.1 Examples

8.1.1 z/OS

To use the ENC_PP program on z/OS, use the following JCL:

```
//ENCPP EXEC PGM=ENC_PP,  
//SYSPRINT DD SYSOUT=*  
//STEPLIB DD DISP=SHR,DSN=+HLQ+.CPTLWARE.MQCE.LOAD
```

9 Appendix D – Capitalware Product Display Version

z/MQCE includes a program to display the product version number.

9.1 Examples

9.1.1 z/OS

To use the CWDSPVER program on z/OS, use the following JCL:

```
//CWDSPVER EXEC PGM=CWDSPVER,  
//SYSPRINT DD SYSOUT=*  
//STEPLIB DD DISP=SHR,DSN=+HLQ+.CPTLWARE.MQCE.LOAD
```


10 Appendix E - Explicitly Setting Values in MSGDATA, RCVDATA & SENDDATA

An alternative to setting values in an IniFile, the user can explicitly set information in channel's MSGDATA, RCVDATA & SENDDATA fields. There are 4 values that can be explicitly set.

- **P** (Perform) indicates what functionality that MQCE will perform. Perform supports 3 values [S / E / B].
- **K** (Key Size) specifies the AES key size used for the encryption / decryption of the message data. Valid values are 128, 192 or 256.
- **D** (Debug Logging) sets the LogMode value to 'D'.
- **L** (License) specifies the queue manager's license key.

```
DEFINE CHANNEL ('MQA1.TO.MQB1') CHLTYPE(SDR) +  
  TRPTYPE(TCP) +  
  XMITQ('MQB1.XMIT') +  
  CONNAME('127.0.0.1(1415)') +  
  MSGEXIT('MQCE') +  
  MSGDATA('P=E;K=256') +  
  REPLACE
```

Note: Separate each value with a semicolon (;).

11 Appendix F – Encryption and Digital Signature

MQ Channel Encryption Solution uses the Advanced Encryption Standard (AES) to encrypt the message data, which flows between IBM MQ (MQ) resources.

Wikipedia

the Advanced Encryption Standard (AES) is an encryption standard adopted by the U.S. government. The standard comprises three block ciphers, AES-128, AES-192 and AES-256, adopted from a larger collection originally published as Rijndael. Each AES cipher has a 128-bit block size, with key sizes of 128, 192 and 256 bits, respectively. The AES ciphers have been analyzed extensively and are now used worldwide, as was the case with its predecessor,[3] the Data Encryption Standard (DES).

AES was announced by National Institute of Standards and Technology (NIST) as U.S. FIPS PUB 197 (FIPS 197) on November 26, 2001 after a 5-year standardization process in which fifteen competing designs were presented and evaluated before Rijndael was selected as the most suitable (see Advanced Encryption Standard process for more details). It became effective as a Federal government standard on May 26, 2002 after approval by the Secretary of Commerce. It is available in many different encryption packages. AES is the first publicly accessible and open cipher approved by the NSA for top secret information

Wikipedia

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

12 Appendix G – Support

The support for MQ Channel Encryption for z/OS can be found at the following location:

By email at:

support@capitalware.com

By regular mail at:

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

13 Appendix H – Summary of Changes

- MQ Channel Encryption for z/OS v3.4.0
 - Enhanced the code for dumping the pointers passed into exit.
 - Fixed an issue in the subroutine that removes trailing blanks
 - Fixed issue when an invalid or expired license key is used
 - Fixed an issue with default exit path

- MQ Channel Encryption for z/OS v3.3.0
 - Tuned the code that is called on entry
 - Tuned the logging code

- MQ Channel Encryption for z/OS v3.2.0
 - Added EncPassPhrase keyword to support the use of encrypted PassPhrase.
 - Added 'ENC_PP' program that will create an encrypted PassPhrase.
 - Addition debug logging information added
 - Fixed an issue in the logging framework where a constant was being modified.

- MQ Channel Encryption for z/OS v3.1.3
 - Addition debug logging information added

- MQ Channel Encryption for z/OS v3.1.1
 - Added check for flags in the CWME header
 - Fixed an issue on Windows with freeing environment variable memory (error with FreeEnvironmentStrings Windows API call)
 - Fixed an issue with using "size_t" variable type when it should have been "int"

- MQ Channel Encryption for z/OS v3.1.0
 - Added keyword 'AllSegments' to signal that all segments are to be encrypted and/or signed.
 - Added extra bounds check for incoming message
 - Removed an extra buffer, hence, reduced memory allocation
 - Improved the IniFile processing speed.
 - Fixed an issue with Enterprise License key not being loaded from a License file.
 - Tested with MQ v8.0

- MQ Channel Encryption for z/OS v3.0.0
 - Altered MQCE header because of incorrect exit data length received from MQ - PMR 58128
 - Tested with MQ v7.1
 - Increased the accepted IniFile parameter length from 1024 to 2048 characters
 - New MQCE Programming Guide manual
 - Fixed a bug in the in-memory Ini parser

- MQ Channel Encryption for z/OS v2.0.0

- Added support for digital signature SHA-2.
- Added program **CWDSPVER** to display the product version number
- MQ Channel Encryption for z/OS v1.0.0
 - Initial release.

14 Appendix I – License Agreement

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

SOFTWARE LICENSE

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

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

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

LIMITED WARRANTY

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

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

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

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

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

15 Appendix J – Notices

Trademarks:

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

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

Intel is a registered trademark of Intel Corporation.

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

Linux is a trademark of Linus Torvalds.

Mac OS X is a trademark of Apple Computer Inc.

Microsoft, Windows, Windows, 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.