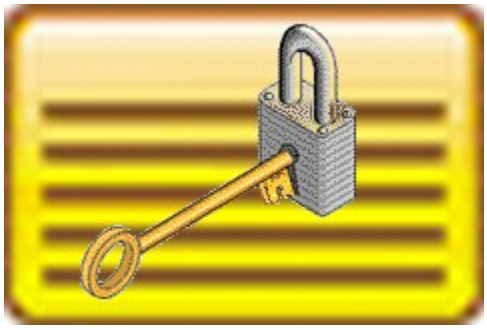
MQ Channel Encryption Programming Guide



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

Last Updated: January 2021.

Table of Contents

1 Introduction

1.1 Overview

MQ Channel Encryption (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.

MQCE provides encryption for message data, which flows between IBM MQ resources. MQCE operates with IBM MQ v7.0, v7.1, v7.5, v8.0, v9.0, v9.1 and v9.2 in Windows, Unix, IBM i (OS/400) and Linux environments. It operates with Sender, Receiver, Server, Requestor, Cluster-Sender, Cluster-Receiver, Server Connection and Client Connection channels of the MQ queue managers.

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.

1. Queue manager to queue manager protection means all messages flowing over a channel between 2 queue managers will be encrypted.

2. Client application to queue manager protection means application-level message data flowing between a MQ client application and queue manager will be encrypted.

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

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.

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

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

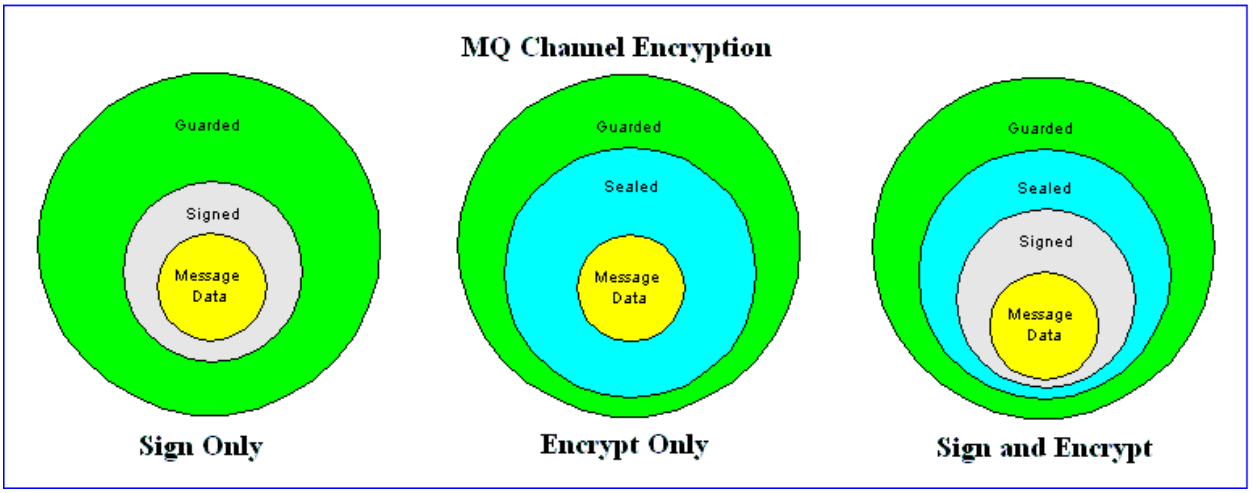
1.2 Executive Summary

The MQCE solution is an MQ encryption exit. It is available for a wide range of platforms: AIX, HP-UX, IBM i, Linux, Solaris and Windows.

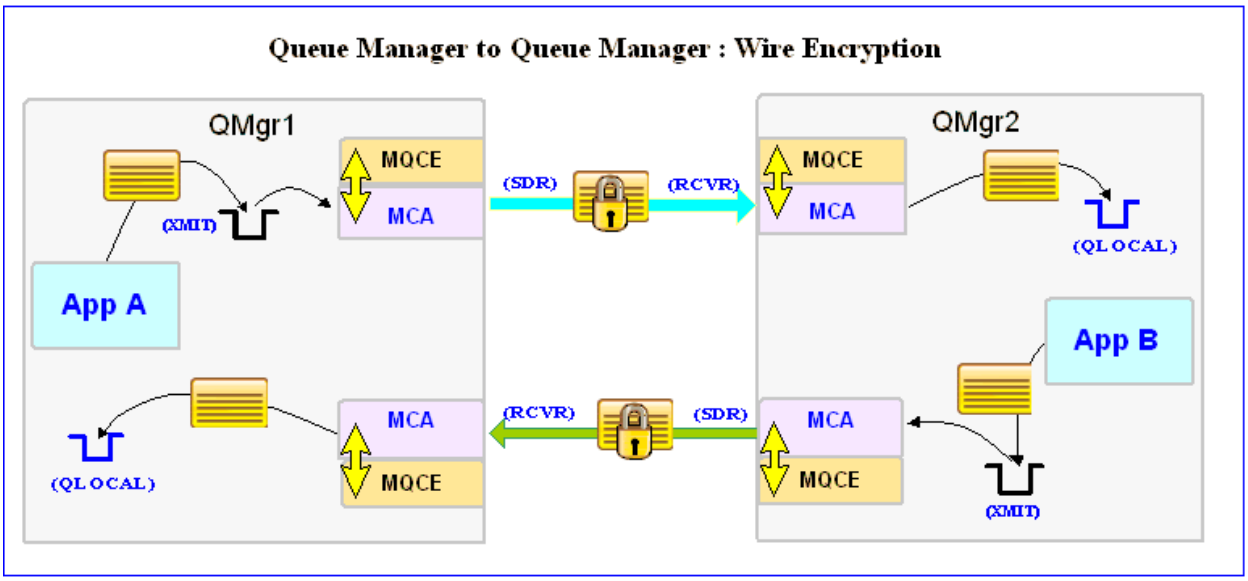
Major Features of MQCE:

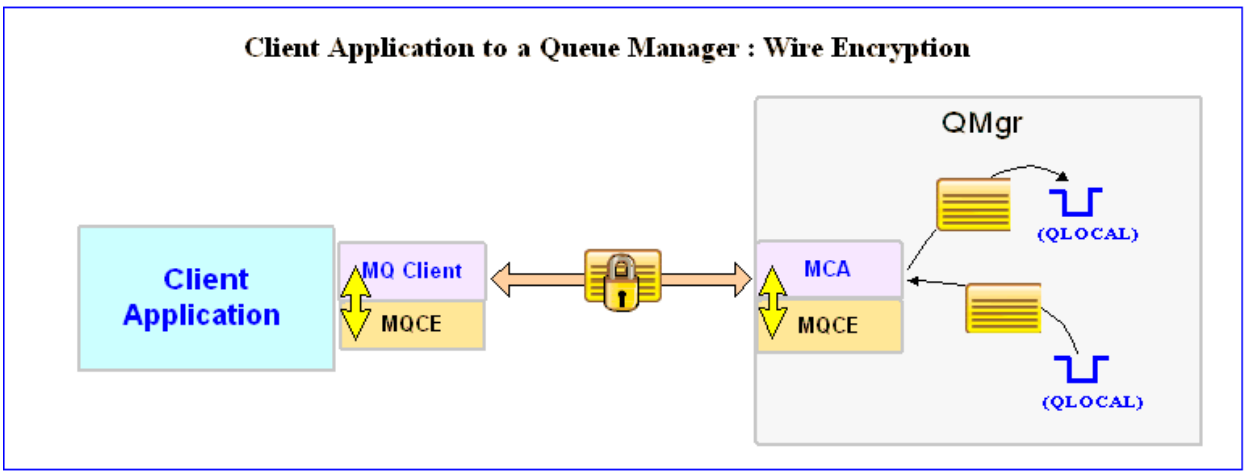
- Easy to set up and configure (unlike SSL)
- No application changes required
- Can be configured as either queue manager to queue manager or client application to queue manager solution
- For both modes, all message data flowing over a channel will be encrypted (nothing missed or forgotten)
- Secure 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 high-level logging capability for encryption / decryption processing

1.3 Message Diagram (Logical View)



1.4 Context Diagram (Logical View)





2 C Language

For C Language, the programmer has 2 different methods to set the MQCE Send/Receive Exit values.

1. If the programmer's application uses the MQCONN API, the user needs to use a CCDT (Client Channel Definition Table) file.

2. If the programmer's application uses the MQCONNX API, the Send/Receive Exit values can be set via MQCONNX API call.

2.1 MQCONN

It is assumed that the user has previously set up an entry in a CCDT file for use by the user's application.

2.1.1 Syntax

MQCONN (QMName, HConn, CompCode, Reason)

2.1.2 Parameters

The MQCONN call has the following parameters as described below: QMName, HConn, CompCode and Reason.

- **QMName (char 48) - input**

The name of the queue manager to which the application wants to connect

- **HConn (MQHCONN) - output**

This handle represents the connection to the queue manager.

- **CompCode (MQLONG) - output**

The completion code of the MQCONN API call

- **Reason (MQLONG) - output**

The reason code of the MQCONN API call

2.1.3 Language Invocations

It is assumed that the user has previously set up an entry in a CCDT file for use by the user's application.

2.1.3.1 C Language

```
MQHCONN    HConn;
MQLONG     CompCode;
MQLONG     Reason;
char       QMName[MQ_Q_MGR_NAME_LENGTH+1];

MQCONN (QMName,
        &HConn,
        &CompCode,
        &Reason);
```

2.1.3.2 Visual Basic Language

```
Dim QMName As String
Dim Hconn As Long
Dim CompCode As Long
Dim Reason As Long

MQCONN QMName, Hconn, CompCode, Reason
```

2.2 MQCONNX

This section describes how to use MQCONNX API to set the MQCE Send/Receive Exit values.

2.2.1 Syntax

MQCONNX (QMName, ConnectOptions, HConn, CompCode, Reason)

2.2.2 Parameters

The MQCONNX call has the following parameters as described below: QMName, ConnectOptions , HConn, CompCode and Reason.

- **QMName (char 48) - input**

The name of the queue manager to which the application wants to connect

- **ConnectOptions (MQHCONN) – input / output**

The ConnectOptions allows the application to specify options relating to the connection to the queue manager.

- **HConn (MQHCONN) - output**

This handle represents the connection to the queue manager.

- **CompCode (MQLONG) - output**

The completion code of the MQCONN API call

- **Reason (MQLONG) - output**

The reason code of the MQCONN API call

2.2.3 Language Invocations

The MQCONNX API call is supported in the following programming languages (C and Visual Basic) as shown below.

2.2.3.1 C Language

```
MQCNO     ConnectOptions = {MQCNO_DEFAULT};
MQCD      ClientConn = {MQCD_CLIENT_CONN_DEFAULT};
MQHCONN   HConn;
MQLONG    CompCode;
MQLONG    Reason;
char      QMName[MQ_Q_MGR_NAME_LENGTH+1];
char      channelName[MQ_CHANNEL_NAME_LENGTH+1];
char      hostname[1024];
char      exitName[1024]= "C:\\Capitalware\\MQCE\\mqce(CE)";

strncpy(ClientConn.ConnectionName,
        hostname,
        MQ_CONN_NAME_LENGTH);

strncpy(ClientConn.ChannelName,
        channelName,
        MQ_CHANNEL_NAME_LENGTH);

strncpy(ClientConn.SendExit,
        exitName,
        MQ_EXIT_NAME_LENGTH);

strncpy(ClientConn.ReceiveExit,
        exitName,
        MQ_EXIT_NAME_LENGTH);

ConnectOptions.ClientConnPtr = &ClientConn;
ConnectOptions.Version = MQCNO_VERSION_6;

MQCONNX(QMName,
        &ConnectOptions,
        &HConn,
        &CompCode,
        &Reason);
```

2.2.3.2 Visual Basic Language

```
Dim CNOCD As MQCNOCD
Dim QMName As String
Dim Hconn As Long
Dim CompCode As Long
Dim Reason As Long

MQCNOCD_DEFAULTS CNOCD

CNOCD.ChannelDef.ConnectionName = GUI_hostname.Text
CNOCD.ChannelDef.ChannelName = GUI_chlName.Text
CNOCD.ChannelDef.Version = MQCD_CURRENT_VERSION
CNOCD.ChannelDef.SendExit = "C:\Capitalware\MQCE\mqce(CE)"
CNOCD.ChannelDef.ReceiveExit = "C:\Capitalware\MQCE\mqce(CE)"

MQCONNXAny QMName, CNOCD, Hconn, CompCode, Reason
```

3 C++ Language

For C++ Language, the programmer has 2 different methods to set the MQCE Send/Receive Exit values.

1. If the programmer's application uses the ImqQueueManager class, the user needs to use a CCDT (Client Channel Definition Table) file.

2. If the programmer's application uses the ImqQueueManager and ImqChannel classes, the Send/Receive Exit values can be set via ImqChannel class.

3.1 ImqQueueManager (MQCONN)

It is assumed that the user has previously set up an entry in a CCDT file for use by the user's application.

3.1.1 Syntax

```
ImqQueueManager     mgr;

mgr.setName( QMName );
```

3.1.2 Parameters

The ImqQueueManager class require the following parameters as described below: QMName.

- **QMName (char 48) - input**

The name of the queue manager to which the application wants to connect

3.1.3 Language Invocations

The ImqQueueManager class is supported in the following programming language (C++) as shown below.

3.1.3.1 C++ Language

```
ImqQueueManager  mgr;
char             QMName[MQ_Q_MGR_NAME_LENGTH+1];

mgr.setName(QMName);

if ( ! mgr.connect( ) )
{
  return( 1 );
}
```

3.2 ImqQueueManager and ImqChannel (MQCONNX)

This section describes how to use the ImqQueueManager and ImqChannel classes to set the MQCE Send/Receive Exit values.

3.2.1 Syntax

```
ImqQueueManager     mgr;
ImqChannel          *pchannel;

mgr.setName( QMName );
pchannel -> setChannelName( ChannelName );
pchannel -> setConnectionName( ConnName );
pchannel -> setSendExitName(exitName);
pchannel -> setReceiveExitName(exitName);
```

3.2.2 Parameters

The ImqQueueManager and ImqChannel classes require the following parameters as described below: QMName, Channelname, ConnName, ExitName and SecurityData.

- **QMName (char 48) - input**

The name of the queue manager to which the application wants to connect

- **ChannelName (char 20) – input**

The name of the channel to use for the connection

- **ConnName (char 264) - input**

The ConnName is the hostname or IP address and Port Number of the remote server where the queue manager is located.

- **ExitName (char 128) – input**

The full path and name of the MQCE send/receive exit

3.2.3 Language Invocations

The ImqQueueManager and ImqChannel classes are supported in the following programming language (C++) as shown below.

3.2.3.1 C++ Language

```
ImqQueueManager  mgr;
ImqChannel      *pchannel = 0;

char      QMName[MQ_Q_MGR_NAME_LENGTH+1];
char      channelName[MQ_CHANNEL_NAME_LENGTH+1];
char      hostname[1024];
char      exitName[1024]="C:\\Capitalware\\MQCE\\mqce(CE)";

mgr.setName(QMName);

pchannel = new ImqChannel ;
pchannel -> setHeartBeatInterval( 1 );
pchannel -> setTransportType( MQXPT_TCP );
pchannel -> setChannelName(channelName);
pchannel -> setConnectionName(hostname);
pchannel -> setSendExitName(exitName);
pchannel -> setReceiveExitName(exitName);
mgr.setChannelReference( pchannel );

if ( ! mgr.connect( ) )
{
  delete pchannel;
  return( 1 );
}
```

4 Java Language

For Java Language, the programmer has 2 different methods to set the MQCE Send/Receive Exit values.

1. If the programmer's application uses the IBM MQ base Java, the MQCEJ class must be used

2. If the programmer's application uses the IBM MQ base JMS, the MQCEJ class must use a QCF (Queue Connection Factory) that contains MQCEJ defined for SENDEXIT and RECEXIT properties.

4.1 IBM MQ base Java

This section describes how to instantiate MQCEJ base Java. There are three ways to instantiate the MQCEJ base Java client-side channel exit.

4.1.1 Syntax

```
new MQCEJ();
new MQCEJ(filename);
new MQCEJ(inlineKeywords);
```

4.1.2 Parameters

The MQCEJ base Java instantiation can include the following parameters as described below: none or filename or inline-keywords.

4.1.2.1 Filename (String) – input

The filename represents the name of the property file (IniFile) that contains the IniFile keyword values.

4.1.2.2 inlineKeywords (String) - input

A string with the IniFile keywords separated by a semi-colon. i.e. K=256

4.1.3 Exceptions

The following exceptions may be encountered:

- IllegalArgumentException
  Invalid / illegal value supplied as an argument to the call.

- FileNotFoundException
  The specified property file (IniFile) could not be found at the location given.

4.1.4 Language Invocations

The MQCEJ base Java only supports the Java programming language.

4.1.4.1 Java Language

Sample #1 does not pass an IniFile or UserId & Password to the MQCEJ client-side channel exit; hence the exit will display a log on pop-up to the end-user.

```
String qManager;
MQEnvironment.hostname = "10.10.10.10(1414)";
MQEnvironment.channel = "TEST.CHL";
MQEnvironment.sendExit = new MQCEJ();
MQEnvironment.receiveExit = new MQCEJ();

MQQueueManager _qMgr = new MQQueueManager(qManager);
```

Sample #2 passes an IniFile to the MQCEJ class. The IniFile contains the UserId and Password that will be used by the MQCEJ client-side channel exit.

```
String qManager;
MQEnvironment.hostname = "10.10.10.10(1414)";
MQEnvironment.channel = "TEST.CHL";
MQEnvironment.sendExit=new MQCEJ("C:\\Capitalware\\MQCE\\mqce.ini");
MQEnvironment.receiveExit=new MQCEJ("C:\\Capitalware\\MQCE\\mqce.ini");

MQQueueManager _qMgr = new MQQueueManager(qManager);
```

Sample #3 passes the UserId and Password directly to the MQCEJ client-side channel exit.

```
String qManager;
String userID;
String password;
MQEnvironment.hostname = "10.10.10.10(1414)";
MQEnvironment.channel = "TEST.CHL";
MQEnvironment.sendExit = new MQCEJ("K=256");
MQEnvironment.receiveExit = new MQCEJ("K=256");

MQQueueManager _qMgr = new MQQueueManager(qManager);
```

4.2 IBM MQ base JMS

This section describes how to use the setSendExit, setReceiveExit, setSendExitInit and setReceiveExitInit methods of the MQQueueConnectionFactory class to set the MQCE Send/Receive Exit values. The setSendExitInit and setReceiveExitInit methods accepts input parameters in 3 different forms: none, filename or inline-keywords.

4.2.1 Syntax

```
MQQueueConnectionFactory mqQCF = new MQQueueConnectionFactory();

mqQCF.setSendExit("biz.capitalware.mqce.MQCEJ");
mqQCF.setSendExitInit(parms);

mqQCF.setReceiveExit("biz.capitalware.mqce.MQCEJ");
mqQCF.setReceiveExitInit(parms);
```

4.2.2 Parameters

The setSendExitInit and setReceiveExitInit methods of the MQQueueConnectionFactory class can include the following parameters as described below: none, filename or inline-keywords.

4.2.2.1 Filename (String) – input

The filename represents the name of the property file (IniFile) that contains the IniFile keyword values.

4.2.2.2 inlineKeywords (String) - input

A string with the IniFile keywords separated by a semi-colon. i.e. K=256

4.2.3 Exceptions

The following exceptions may be encountered:

- IllegalArgumentException
  Invalid / illegal value supplied as an argument to the call.

- FileNotFoundException
  The specified property file (IniFile) could not be found at the location given.

4.2.4 Language Invocations

The MQCEJ for JMS only supports the Java/JMS programming language.

4.2.4.1 Java/JMS Language

Sample #1 uses a QCF via an MQJNDI entry. The QCF entry includes the definition for the MQCEJ send and receive exits.

```
QueueConnectionFactory qcf;
QueueConnection connection;
Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY, JNDI_CONTEXT);
env.put(Context.PROVIDER_URL, "file:/C:\JNDI\test\mqjndi");

Context ctx = new InitialContext(env);
qcf = (QueueConnectionFactory) ctx.lookup(myQCF);
connection = qcf.createQueueConnection();
```

Sample #2 uses a dynamically created QCF. The programmer must explicitly set the MQCEJ send and receive exits via the setSendExit and setReceiveExit methods of the QCF.

```
MQQueueConnectionFactory mqQCF;
QueueConnection connection;
String qManager;

mqQCF = new MQQueueConnectionFactory();
mqQCF.setQueueManager(qManager);
mqQCF.setHostName("10.10.10.10(1414)");
mqQCF.setChannel("TEST.CHL");
mqQCF.setTransportType(JMSC.MQJMS_TP_CLIENT_MQ_TCPIP);
mqQCF.setSendExit("biz.capitalware.mqce.MQCEJ");
mqQCF.setSendExitInit("K=256");
mqQCF.setReceiveExit("biz.capitalware.mqce.MQCEJ");
mqQCF.setReceiveExitInit("K=256");

connection = mqQCF.createQueueConnection();
```

5 .NET C-Sharp Language

For the .NET C-Sharp Language, the programmer has 2 different methods to set the MQCE Send/Receive Exit values. One method uses the new MQCEDN .NET class under a managed .NET environment and the other method uses the native mqce.dll under an unmanaged .NET environment.

1. If the programmer's application uses a managed .NET environment, the MQCEDN class must be used

2. If the programmer's application uses an unmanaged .NET environment, the native mqce.dll must be used

5.1 Managed .NET Environment

This section describes how to instantiate MQCEDN class.

5.1.1 Syntax

```
MQEnvironment.SendExit="C:\\Capitalware\\MQCE\\mqcedn.dll(Capitalware.MQCEDN)";
MQEnvironment.ReceiveExit="C:\\Capitalware\\MQCE\\mqcedn.dll(Capitalware.MQCEDN)";
```

5.1.2 Parameters

There are no parameters for the MQCEDN class.

5.1.3 Exceptions

There are no MQCEDN exceptions.

5.1.4 Language Invocations

The MQCEDN class supports any managed .NET language (e.g. C-Sharp .NET and VB.NET).

5.1.4.1 C-Sharp Language

```
String qManager;
MQEnvironment.Hostname = "10.10.10.10(1414)";
MQEnvironment.Channel = "TEST.CHL";
MQEnvironment.SendExit="C:\\Capitalware\\MQCE\\mqcedn.dll(Capitalware.MQCEDN)";
MQEnvironment.ReceiveExit="C:\\Capitalware\\MQCE\\mqcedn.dll(Capitalware.MQCEDN)";

MQQueueManager _qMgr = new MQQueueManager(qManager);
```

6 Appendix A – Sample Client Channel Table

The following are sample Client Channel Table entries that can be used with the sample code for MQCONN (ImqQueueManager), CWMQCONN or MQQueueManager (see Appendix C for sample code).

6.1 Windows

```
DEFINE CHANNEL('TEST.CHL') CHLTYPE(CLNTCONN) +
   TRPTYPE(TCP) CONNAME('10.10.10.10(1414)') QMNAME('MQA1') +
   SENDDATA(' ') SENDEXIT('C:\Capitalware\MQCE\mqce(CE)') +
   RCVDATA(' ') RCVEXIT('C:\Capitalware\MQCE\mqce(CE)')
```

6.2 Unix and Linux for IBM MQ 32-bit

```
DEFINE CHANNEL('TEST.CHL') CHLTYPE(CLNTCONN) +
   TRPTYPE(TCP) CONNAME('10.10.10.10(1414)') QMNAME('MQA1') +
   SENDDATA(' ') SENDEXIT('/var/mqm/exits/mqce(CE)') +
   RCVDATA(' ') RCVEXIT('/var/mqm/exits/mqce(CE)')
```

6.3 Unix and Linux for IBM MQ 64-bit

```
DEFINE CHANNEL('TEST.CHL') CHLTYPE(CLNTCONN) +
   TRPTYPE(TCP) CONNAME('10.10.10.10(1414)') QMNAME('MQA1') +
   SENDDATA(' ') SENDEXIT('/var/mqm/exits64/mqce(CE)') +
   RCVDATA(' ') RCVEXIT('/var/mqm/exits/mqce(CE)')
```

6.4 Java Applications

```
DEFINE CHANNEL('TEST.CHL') CHLTYPE(CLNTCONN) +
   TRPTYPE(TCP) CONNAME('10.10.10.10(1414)') QMNAME('MQA1') +
   SENDDATA(' ') SENDEXIT('biz.capitalware.mqce.MQCEJ') +
   RCVDATA(' ') RCVEXIT('biz.capitalware.mqce.MQCEJ')
```

7 Appendix B – Sample MQJNDI

The following are sample MQJNDI entries that can be used by the Java/JMS code samples (see Appendix C for sample code):

7.1 JMS Queue Connection Factory (QCF) Sample:

```
DEFINE QCF(myQCF) QMANAGER(MQA1) CHANNEL(TEST.CHL)
       HOSTNAME(10.10.10.10) PORT(1414)
       SENDEXIT(biz.capitalware.mqce.MQCEJ)
       SENDEXITINIT('')
       RECEXIT(biz.capitalware.mqce.MQCEJ)
       RECEXITINIT('')
       FAILIFQUIESCE(YES) TRANSPORT(CLIENT)
```

7.2 JMS Queue Sample:

```
DEFINE Q(mqs.test.q) QUEUE(TEST.Q1) QMANAGER(MQA1)
       TARGCLIENT(JMS) FAILIFQUIESCE(YES)
```

8 Appendix C – MQCE Language Files

The following is the directory structure layout followed by the Language files:

Windows Directory Structure	Unix Directory Structure
C: +--Capitalware +--MQCE <- Install Directory +--samples +--c +--cpp +--cs +--java +--vb	<Install_Directory> +--Capitalware +--MQCE +--samples +--c +--cpp +--java

8.1 MQCE C Sample Files

The MQCE C sample files are installed in the following directories:

Platform	Directory
Linux / Unix	<Install_Directory>/samples/c/
Windows	C:\Capitalware\MQCE\samples\c\

8.1.1 List of C sample files

Filename	Description
MQCETest01.c	Demonstrates how to use the MQCONN and CCDT to connect to a queue manager then how to open a queue, ***put*** a message to a queue, close the queue and disconnect from a queue manager.
MQCETest02.c	Demonstrates how to use the C MQCONN and CCDT to connect to a queue manager then how to open a queue, ***get*** a message from a queue, close the queue and disconnect from a queue manager.
MQCETest11.c	Demonstrates how to use the MQCONNX API with the MQCE send/receive exit to connect to a queue manager then how to open a queue, ***put*** a message to a queue, close the queue and disconnect from a queue manager.
MQCETest12.c	Demonstrates how to use the MQCONNX API with the MQCE send/receive exit to connect to a queue manager then how to open a queue, ***get*** a message from a queue, close the queue and disconnect from a queue manager.

8.2 MQCE C++ Sample Files

The MQCE C sample files are installed in the following directories:

Platform	Directory
Linux / Unix	<Install_Directory>/samples/cpp/
Windows	C:\Capitalware\MQCE\samples\cpp\

8.2.1 List of C++ sample files

Filename	Description
MQCETest01.cpp	Demonstrates how to use the ImqQueueManager class with CCDT to connect to a queue manager then how to open a queue, ***put*** a message to a queue, close the queue and disconnect from a queue manager.
MQCETest02.cpp	Demonstrates how to use the ImqQueueManager class with CCDT to connect to a queue manager then how to open a queue, ***get*** a message from a queue, close the queue and disconnect from a queue manager.
MQCETest11.cpp	Demonstrates how to use the ImqQueueManager and ImqChannel classes with the MQCE send/receive exit to connect to a queue manager then how to open a queue, ***put*** a message to a queue, close the queue and disconnect from a queue manager.
MQCETest12.cpp	Demonstrates how to use the ImqQueueManager and ImqChannel classes with the MQCE send/receive exit to connect to a queue manager then how to open a queue, ***get*** a message from a queue, close the queue and disconnect from a queue manager.

8.3 MQCE base Java & JMS Sample Files

The MQCE base Java and JMS sample files are installed in the following directories:

Platform	Directory
Linux / Unix	<Install_Directory>/samples/java/
Windows	C:\Capitalware\MQCE\samples\java\

8.3.1 List of Java sample files

Filename	Description
MQCETest01.java	Demonstrates how to use the MQQueueManager, MQEnvironment and MQCEJ class to connect to a queue manager then how to open a queue, ***put*** a message to a queue, close the queue and disconnect from a queue manager.
MQCETest02.java	Demonstrates how to use the MQQueueManager, MQEnvironment and MQCEJ class to connect to a queue manager then how to open a queue, ***get*** a message from a queue, close the queue and disconnect from a queue manager.
MQCETest11.java	Demonstrates how to use the MQQueueManager, HashTable and MQCEJ class to connect to a queue manager then how to open a queue, ***put*** a message to a queue, close the queue and disconnect from a queue manager.
MQCETest12.java	Demonstrates how to use the MQQueueManager, HashTable and MQCEJ class to connect to a queue manager then how to open a queue, ***get*** a message from a queue, close the queue and disconnect from a queue manager.
MQCETest41.java	Demonstrates how to use the MQQueueManager class with a Client Channel Definition Table to connect to a queue manager then how to open a queue, ***put*** a message to a queue, close the queue and disconnect from a queue manager.
MQCETest42.java	Demonstrates how to use the MQQueueManager class with a Client Channel Definition Table to connect to a queue manager then how to open a queue, ***get*** a message from a queue, close the queue and disconnect from a queue manager.

8.3.2 List of Java/JMS sample files

Filename	Description
MQCETestJMS01.java	Demonstrates how to use the QueueConnectionFactory (QCF) via MQJNDI and MQCEJ class to connect to a queue manager then how to open a queue, ***put*** a message to a queue, close the queue and disconnect from a queue manager.
MQCETestJMS02.java	Demonstrates how to use the QueueConnectionFactory (QCF) via MQJNDI and MQCEJ class to connect to a queue manager then how to open a queue, ***get*** a message from a queue, close the queue and disconnect from a queue manager.

Filename	Description
MQCETestJMS11.java	Demonstrates how to use the QueueConnectionFactory (QCF) and MQCEJ class to connect to a queue manager then how to open a queue, ***put*** a message to a queue, close the queue and disconnect from a queue manager.
MQCETestJMS12.java	Demonstrates how to use the QueueConnectionFactory (QCF) and MQCEJ class to connect to a queue manager then how to open a queue, ***get*** a message from a queue, close the queue and disconnect from a queue manager.

8.4 .NET C-Sharp Sample Files

The MQCE .NET C-Sharp sample files are installed in the following directories:

Platform	Directory
Windows	C:\Capitalware\MQCE\samples\cs\

8.4.1 List of .NET C-Sharp sample files

Filename	Description
MQCETest01.cs	Demonstrates how to use the MQQueueManager, MQEnvironment and MQCEDN class to connect to a queue manager then how to open a queue, ***put*** a message to a queue, close the queue and disconnect from a queue manager.
MQCETest02.cs	Demonstrates how to use the MQQueueManager, MQEnvironment and MQCEDN class to connect to a queue manager then how to open a queue, ***get*** a message from a queue, close the queue and disconnect from a queue manager.
MQCETest11.cs	Demonstrates how to use the MQQueueManager, MQEnvironment and MQCEDN class to connect to a queue manager then how to open a queue, ***put*** a message to a queue, close the queue and disconnect from a queue manager.
MQCETest12.cs	Demonstrates how to use the MQQueueManager, MQEnvironment and MQCEDN class to connect to a queue manager then how to open a queue, ***get*** a message from a queue, close the queue and disconnect from a queue manager.
MQCETest41.cs	Demonstrates how to use the MQQueueManager class (unmanaged .NET) with a Client Channel Table to connect to a queue manager then how to open a queue, ***put*** a message to a queue, close the queue and disconnect from a queue manager.
MQCETest42.cs	Demonstrates how to use the MQQueueManager class (unmanaged .NET) with a Client Channel Table to connect to a queue manager then how to open a queue, ***get*** a message from a queue, close the queue and disconnect from a queue manager.

8.5 MQCE Visual Basic Sample Files

The MQCE Visual Basic sample files are installed in the following directories:

Platform	Directory
Windows	C:\Capitalware\MQCE\samples\vb\

8.5.1 List of Visual Basic sample files

Filename	Description
MQCETest01.frm	Demonstrates how to use the MQCONN and CCDT to connect to a queue manager then how to open a queue, ***put*** a message to a queue, close the queue and disconnect from a queue manager.
MQCETest02.frm	Demonstrates how to use the MQCONN and CCDT to connect to a queue manager then how to open a queue, ***get*** a message from a queue, close the queue and disconnect from a queue manager.
MQCETest11.frm	Demonstrates how to use the MQCONNX API with the MQCE client-side channel exit to connect to a queue manager then how to open a queue, ***put*** a message to a queue, close the queue and disconnect from a queue manager.
MQCETest12.frm	Demonstrates how to use the MQCONNX API with the MQCE client-side channel exit to connect to a queue manager then how to open a queue, ***get*** a message from a queue, close the queue and disconnect from a queue manager.

9 Appendix D – 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 user, or group of users for Muliple User Licenses and Site Licenses. This Agreement requires that each user of the SOFTWARE be Licensed, either individually, or as part of a group. A Multi-User License provides for a specified number of users to use this SOFTWARE at any time. This does not provide for concurrent user Licensing. Each user of this SOFTWARE must be covered either individually, or as part of a group Multi-User License. The SOFTWARE is in use on a computer when it is loaded into the temporary memory (i.e. RAM) or installed into the permanent memory (e.g. hard disk) of that computer. This software may be installed on a network provided that appropriate restrictions are in place limiting the use to registered users only.

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 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.

10 Appendix E – 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, Visual Basic, 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.