Kofax Communication Server

TC/LINK-MQ Technical Manual

Version: 10.2.0

CONTENTS

1. Overview

IBM WebSphere MQ is a widely deployed messaging middleware that connects various platforms with a reliable messaging mechanism.

IBM WebSphere MQ runs on a huge number of platforms, including MVS/ESA, OS/400, AIX, Sun Solaris, HP-UX, SINIX, OS/2, Windows and Tandem.

Main idea behind IBM WebSphere MQ is to provide application-to-application connectivity. IBM WebSphere MQ is deployed in enterprises to automate communication processes that stretch over different environments, like an MVS host application talking to an R/3 system under UNIX.

With TC/LINK-MQ, Kofax Communication Server immediately provides its XML Message Format or the TC OPEN MESSAGE FORMAT (TOM) on all platforms where IBM WebSphere MQ is available. This means that customers can easily adapt their applications on all their platforms so that they talk to Kofax Communication Server systems in a simple and unified way.

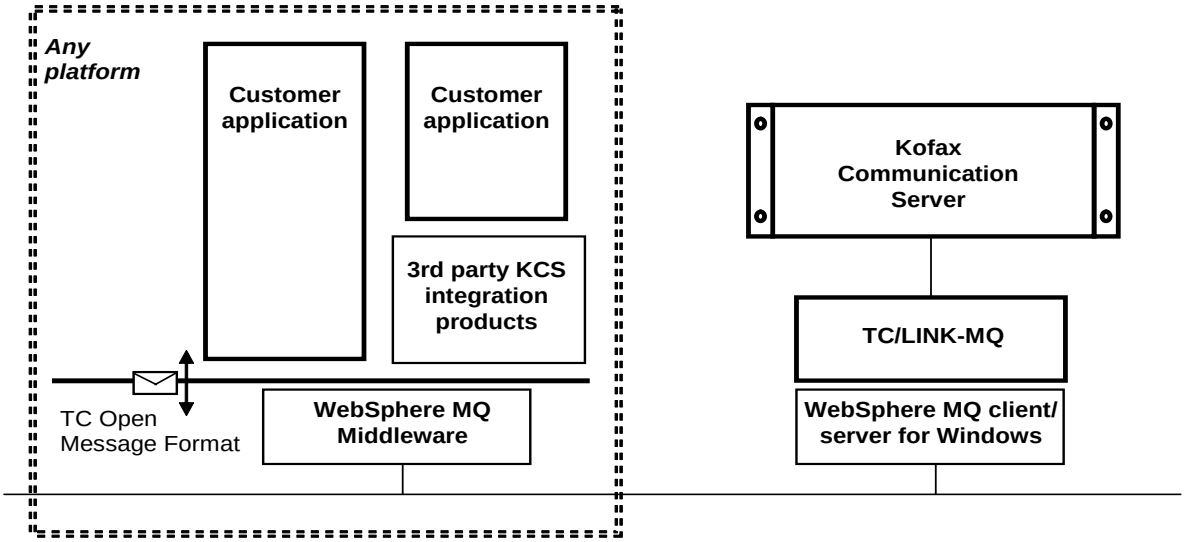
In addition to the TC Open Message Interface provided by TC/LINK-MQ, host integration products will become available, providing specific integration with various environments. These products will be made available by Kofax Solution Providers.

1.1 TC/LINK-MQ Positioning, Advantage, Strengths

TC/LINK-MQ provides:

- KCS connectivity for all platforms supported by IBM WebSphere MQ
- Full store-and-forward functionality (send/receive, attachments and notifications)
- Directory synchronization into KCS allows leveraging of full KCS functionality like inbound routing, message templates, access control and use of KCS proxy addresses
- Messages are transferred via standard IBM WebSphere MQ mechanism in XML or the simple and flexible TC OPEN MESSAGE FORMAT
- Can be used from any host platform with any programming language supported by IBM WebSphere MQ
- Open, well-documented interface makes it easy to connect existing applications. In addition Partner 'host integration' products will become available, providing an even higher level of integration on different host platforms.
- Based on proven KCS TC/LINK architecture
- Full support of all TC/LINK features including document conversion, load balancing, fail-over, SNMP, etc.

1.2 Structure of TC/LINK-MQ



TC/LINK-MQ connects to any supported platform via a locally installed IBM WebSphere MQ server or client. Connection to the customer application can be established via the most common network protocols (TCP/IP, NetBIOS, SPX, or LU6.2).

1.3 Support of IBM WebSphere

IBM MQSeries is part of the WebSphere product family. TC/LINK-MQ supports the following versions:

- IBM WebSphere MQ 7.0
- IBM WebSphere MQ 7.1
- IBM WebSphere MQ 7.5
- IBM WebSphere MQ 8.0
- IBM WebSphere MQ 8.2

1.3.1 Note on MQ 7.1 or later

Since MQ 7.1 there is a new queue manager parameter: CHLAUTH. By default this parameter is enabled and you might receive authorization errors. You can disable this feature, if desired, by entering the following command from runmqsc:

```
ALTER QMGR CHLAUTH(DISABLED)
```

1.3.2 Note on MQ 8.0 or later

Since MQ 8.0 there is an additional queue manager parameter: CONNAUTH. By default this parameter is enabled and you might receive authorization errors. You can disable this feature, if desired, by entering the following command from runmqsc:

```
ALTER QMGR CONNAUTH('')
```

AMQCLCHL.TAB
The AMQCLCHL.TAB file location changed to the ProgramData part of the folder structure.

MQ Server (where the file is created):

```
C:\ProgramData\IBM\MQ\qmgrs\<QueueManagerName>\@ipcc\AMQCLCHL.TAB
```

MQ Client (where the file from the server has to be copied to):

```
C:\ProgramData\IBM\MQ
```

2. IBM WebSphere MQ Mail Background

This section will give you a quick overview of the basic IBM WebSphere MQ functionality needed for understanding TC/LINK-MQ. This section is restricted to IBM WebSphere MQ on Windows; if you need to know more about different platforms, special features of IBM WebSphere MQ or similar, refer to the IBM documentation!

2.1 Basic IBM WebSphere MQ Functionality

WebSphere MQ – as the name “Message queuing” already says – is a product that transports data between applications by use of queues.

One application puts data (so-called “MQ messages”) on a special queue, while another application gets the data out again. These applications may be on the same machine, on different machines, or even on different hardware platforms on the different ends of the world.

If configured correctly, IBM WebSphere MQ guarantees that the data put into a queue is delivered “once – and only once” to the intended recipient application. If the recipient application is temporarily unreachable, the data is buffered; retries will be done as configured.

Applications using IBM WebSphere MQ don’t need to care about the network transport; accessing WebSphere MQ is the same for all physical link types, and network protocols.

2.2 Important Objects for TC/LINK-MQ

To move data between applications and TC/LINK-MQ, some IBM WebSphere MQ objects are required.

2.2.1 Queue Manager

A queue manager controls the operation of one or more queues and channels. Before you can access the queues, the queue manager needs to be started (in fact, the queue manager is usually started at system startup, and keeps running as a service all the time).

The most important properties of a queue manager are

- Maximum message size allowed (MAXMSGL) … the maximum message size that can actually be used is the minimum of this value and the corresponding queue property.
- Message log settings … must be large enough to enable large persistent messages (see 5.5.1 “Large Message Operation” for details)
- Default character set ID (CCSID): is usually 850 on Windows.

2.2.2 Local Queues

As already mentioned, queues are the place where messages are put to/get from. In local queues, the messages are buffered until they are delivered to the next queue (if the messages go through “channels” – see below), or they are finally retrieved from the receiving application.

The most important queue properties are

- Maximum message size allowed (MAXMSGL)
- Maximum queue depth allowed (MAXDEPTH) (see also 8.1.6 “Maximum Queue Depth”)
- Current queue depth (CURDEPTH)
- Default persistence (DEFPSIST); this gives a default for message persistence. This property describes – in short – whether messages shall survive queue manager restarts (persistent) or not (not persistent).
- Usage (this may be either NORMAL … a simple local queue, or XMITQ … queue is used for message transmission to a different queue manager)

2.2.3 IBM WebSphere MQ Messages

The data on IBM WebSphere MQ queues is exchanged in so-called “messages”. One application puts messages to the queue, identifying the destination queue; IBM WebSphere MQ delivers the message to the desired destination queue (if possible); and the destination application gets the message from the queue.

Most important IBM WebSphere MQ message properties are:

- Message size
- Message Identifier
- Correlation Identifier
- Character set ID (CCSID)
- Persistence (if a message is persistent, it survives queue manager restarts or power failure; if not, it will be lost)

2.2.4 Channels

Channels are responsible for interconnection of queues and applications.

There are a few types of channels, and lot of channel properties (the so-called “channel configuration”) that make them very versatile. The most important types are

2.2.4.1 Server Connection Channel (SVRCONN)

This is the channel configuration required to allow IBM WebSphere MQ clients to access all local queues. The client needs to know the name of the channel, plus the network address of the IBM WebSphere MQ

server, put this to the MQSERVER environment variable (see 5.4.2.4.2 “Configuration Using the MQSERVER Environment Variable”) – and then simply connect to the server.

Note:

- An active listener program is required on the server (<runmqlsr>; see 2.4.4 “Starting / Stopping a Listener”)
- Special handling may be required if multiple listeners shall be run on the same server; see the “WebSphere MQ Intercommunication” manual for details.

2.2.4.2 Sender Channel (SDR)

A sender channel is used for actively sending messages to a different IBM WebSphere MQ server. The intended recipient must have a corresponding Receiver channel of the same name to make the transmission work.

With the sender channel, you need to specify the connection data – e.g. destination IP address, and the queue holding the data to be sent – the transmission queue.

Note:

- Per default, sender channels must be manually started (e.g. by the <runmqchl> command). However, you can define an enhanced mechanism called “triggering”; this will start channels automatically as soon as certain conditions are met (e.g. more than 10 messages on the transmission queue). See the “WebSphere MQ Intercommunication” manual for details!
- You can configure several channel parameters, like a inactivity disconnect interval, retries on connection failure, and much more. See the “WebSphere MQ Command Reference” manual for details!

2.2.4.3 Receiver Channel (RCVR)

Is the counterpart of the Sender channel. By default, the receiver channel is started automatically as soon as an incoming connection from the matching sender channel is detected.

2.2.5 Remote Queues

When a message shall move to a destination not on the local queue manager, it needs to know where to go to, and which route to take. This is configured by use of remote queue definitions.

In remote queue definitions, you need to configure the remote queue manager name, plus the queue the message shall be put there. Also required is the name of a transmission queue; this is a local queue that is configured to send messages to a remote queue manager via an associated channel. Finally, also the default persistence setting of the remote queue definition is important: If it is set to YES (recommended!), messages from TC/LINK-MQ will survive queue manager restarts.

See 2.5 “Connecting to Other IBM WebSphere MQ Servers” for an example of using remote queues; this will illustrate the use of remote queues much better than many more words …

2.3 Rules for Naming IBM WebSphere MQ Objects

(The following is an excerpt from the “WebSphere MQ Command Reference” Manual)

The character set that can be used for naming all IBM WebSphere MQ objects is as follows:

- Uppercase A-Z
- Lowercase a-z (however, on systems using EBCDIC Katakana you cannot use lowercase characters, and there are also restrictions on the use of lowercase letters for MVS console support)
- Numerics 0-9
- Period (.)
- Forward slash (/)
- Underscore (_)
- Percent sign (%). The percent sign (%) is a special character to RACF. If you are using RACF as the external security manager for IBM WebSphere MQ for MVS/ESA, you should not use % in object

names. If you do, these names are not included in any security checks when RACF generic profiles are used.

Note:

- The above specification implies that you are not allowed to use any Kanji characters or Umlauts in object names!
- Names in IBM WebSphere MQ are case sensitive; however, you should remember that lowercase characters that are not contained within quotation marks are folded to uppercase.
- Leading or embedded blanks are not allowed.
- You should avoid using names with leading or trailing underscores, because they cannot be handled by the IBM WebSphere MQ for MVS/ESA operations and control panels.
- Any name that is less than the full field length can be padded to the right with blanks. All short names that are returned by the queue manager are always padded to the right with blanks.
- Any structure to the names (for example, the use of the period or underscore) is not significant to the queue manager.

2.4 Basic IBM WebSphere MQ Administration

This section gives a very short excerpt of the “WebSphere MQ Administration” and the “WebSphere MQ Command Reference” manuals. For more detailed information, read on there!

2.4.1 Creating / Deleting Queue Managers

The basic command to create a queue manager is

```
crtmqm <Queue Manager name>
```

You may delete a queue manager by

```
dltmqm <Queue Manager name>
```

Note: All messages that are currently in queues on that queue manager are lost when deleting the queue manager! Therefore, use this command with extreme care!

2.4.1.1 Detail: Queue Manager Log Files

As optional parameters in the <crtmqm> command, you can specify the number/size of log files the new queue manager should have.

What are these log files used for?

- The queue manager uses log files for intermediate data storage. For example, if a persistent message is put to IBM WebSphere MQ, it is written to the log file, before it is actually put to the destination queue. This makes persistent messages surviving power failure at any time of operation.

What is the size of a log file?

- The size of a log file is calculated in pages of 4kByte. So, a “crtmqm -lf 256” parameter will set the log file size to 1Mbyte each (which is the default in IBM WebSphere MQ for Windows).

What is the difference between primary and secondary log files?

- The configured number of primary log files is created (in full size!) at the time the queue manager is started. So, the disk space used by them is always occupied. The secondary log files are created on demand, up to the configured maximum number, and removed if they are no longer needed (after some timeout).
  So, primary log files are faster, but take more disk resources.

Do I have to deal with log file settings when installing TC/LINK-MQ?

- Usually, the default values specified will be sufficient. Only in case of large-message operation, it may be required to increase the number and/or size of the IBM WebSphere MQ log files. See 5.5.1 “Large Message Operation” for details.

Can I change the log file settings after creation of the queue manager?

- You can change the log file counts (NOT the log file size!) by editing the <qm.ini> file of the queue manager (located in the c:\mqm\qmgrs\<queue manager name> directory on Windows). You need to restart the queue manager to make the changes effective. Please be careful to enter correct values (see IBM WebSphere MQ documentation for the corresponding platform)!

Where do I find those log files?

- On the IBM WebSphere MQ server for Windows, the log files are by default located in "c:\mqm\logs\<queue manager name>". On different platforms, the location is different (and also configurable!); so, check the IBM documentation for the platform in question.

2.4.2 Starting / Stopping IBM WebSphere MQ

There are two basic possibilities for starting or stopping IBM WebSphere MQ:

2.4.2.1 From the Command Line

To start a queue manager, type

```
strmqm <Queue Manager name>
```

To stop a queue manager, use the command

```
endmqm <Queue Manager name>
```

If you want an immediate shutdown, you may add the –i option ("immediately"):

```
endmqm –i <Queue Manager name>
```

2.4.2.2 From the IBM WebSphere MQ Service

If you have assigned a queue manager to the IBM WebSphere MQ service (<scmmqm> command; see 5.4.1.4 "What Must Be Configured for Automatic Windows Startup" for details), then you can start/stop the IBM WebSphere MQ server from the Windows Control Panel/Services tab.

Alternatively, you can type at the command line

```
net start IBMMqSeries … to start the server,
net stop IBMMqSeries … to stop it.
```

Note:

- To make use of starting/stopping IBM WebSphere MQ via Windows service, you need to create/adapt a startup command file (always called "Mqstartup.cmd" in our examples), and register it (<scmmqm>) for use with the service. See 5.4.1.4 "What Must Be Configured for Automatic Windows Startup" for details!
- A minimum startup command file, plus the required registration, is automatically created at first TC/LINK-MQ startup (if "create mail dependencies" is enabled).

2.4.3 Creating / Deleting IBM WebSphere MQ Objects / the runmqsc Tool

The most important tool for IBM WebSphere MQ administration is the "WebSphere MQ Command Tool" (runmqsc). It allows access to most properties of the queue manager, queues and channels.

To start the tool, type "runmqsc <Queue manager name>" from the command line. If the queue manager is running, you can then enter IBM WebSphere MQ commands. A few examples are listed in the table below:

Command	Description	Remarks
DEFINE QLOCAL(QUEUENAME)	Creates a local queue of given name	
DEFINE CHANNEL(CHLNAME)	Creates a new channel	Some more mandatory parameters required
ALTER QLOCAL MAXMSGL(100)	Changes a property of the given queue	
START CHANNEL(CHLNAME)	Starts the given channel	
STOP CHANNEL (CHLNAME)	Ends the given channel	

DISPLAY QLOCAL(*) CURDEPTH	Displays the current depth of all local queues	
CLEAR QLOCAL(QUEUENAME)	Clears the content of the given queue; no reports are generated, the messages are simply lost!	This command silently deletes all messages on the queue; therefore, use with extreme care!

To check for all possible parameters or functions, you may either type the intended command with a question mark (e.g. ALTER QLOCAL ?) to get a list of possible options; or, check the “WebSphere MQ Command Reference Manual” for detailed descriptions!

2.4.4 Starting / Stopping a Listener

The listener process is required on Windows to accept connections from outside the local machine.

To start a listener for transport type TCP, type

```
runmqlsr –t tcp –m <Queue Manager Name>
```

To stop it, use

```
endmqlsr –m <Queue Manager Name>
```

You may also add the <runmqlsr> command to the Mqstartup.cmd file; the listener will then automatically start as soon as the IBM WebSphere MQ Windows service is started.

2.4.5 The Channel Initiator

If you use enhanced channel startup techniques like triggering, you need to start a channel initiator process. The command is like

```
runmqchi –m <Queue Manager Name> –q SYSTEM.CHANNEL.INITQ
```

Any time a trigger condition is met by one of the sender channels, a trigger message is placed on the specified trigger queue (default: <SYSTEM.CHANNEL.INITQ>). The <runmqchi> process detects the trigger message, and starts the channel that is assigned to this message.

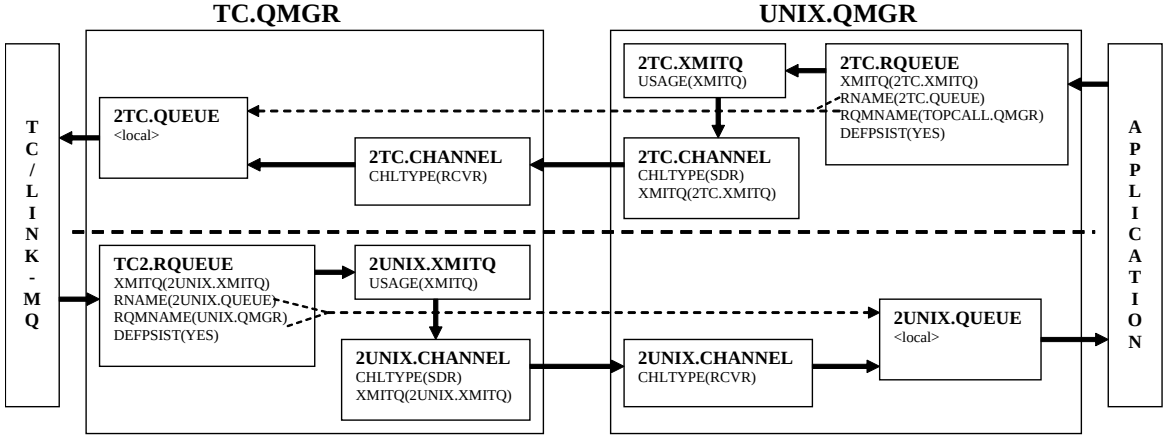
2.5 Connecting to Other IBM WebSphere MQ Servers

In order to make more than one IBM WebSphere MQ server communicate, you have several possibilities. The following lists just a very simple example (yes, really simple! You can add much more complexity); see the “WebSphere MQ Intercommunication” manual for more complex scenarios!

Scenario:

We have TC/LINK-MQ with a local IBM WebSphere MQ server. The application connecting to KCS via TC/LINK-MQ is located on a different IBM WebSphere MQ server (e.g. on UNIX).

We want to set up communication between the two queue managers involved: <TC.QMGR> on the TC/LINK-MQ side, <UNIX.QMGR> on the connected application.



See 8.9 “Advanced Configuration for Server-Server Communication” for an example configuration file of this scenario!

How the messages move in detail:

- A message is sent from KCS via TC/LINK-MQ to IBM WebSphere MQ.
- TC/LINK-MQ will put the message to the <TC2.RQUEUE>
- IBM WebSphere MQ finds that this is a remote queue definition, and puts the message to the defined transmission queue <2UNIX.XMITQ> (the XMITQ parameter). The remote queue manager and queue name are inserted to an internal IBM WebSphere MQ transmission header.
- The sender channel <2UNIX.CHANNEL> will pick the message from its configured transmission queue, and transfer it to the computer defined in its CONNAME parameter (not shown in the graphic)
- The remote queue manager must have a receiver channel of the same name as the sender channel. The receiver channel <2UNIX.CHANNEL> will put the message to the queue indicated in the internal IBM WebSphere MQ transmission header (the RNAME, RQMNAME from the <TC2.RQUEUE> remote queue definition). In our example, this is the <2UNIX.QUEUE>.
- The application that polls the <2UNIX.QUEUE> will finally find and process the message.
- In the other direction (application to TC/LINK-MQ), it is quite the same … just the names are different.

Note:

- Sender channels need to be started in order to transfer messages! This can be done e.g. from the command line with

  ```
  runmqchl –m <Queue Manager Name> -c <Channel Name>,
  ```

  or from the <runmqsc> tool by

  ```
  START CHANNEL <Channel Name>.
  ```

- For advanced possibilities to start sender channels, e.g. starting when a message is put on a queue, see the keyword “triggering” in the “WebSphere MQ Intercommunication” manual.
- Receiver channels are started automatically on demand.
- Some platforms (e.g. Windows) require a listener running on the receiver side. See 2.4.4 ”Starting / Stopping a Listener” for a Windows-based description.
- When stopping a channel, make sure that the associated transmission queue is empty (first stop TC/LINK-MQ, then stop the transmission channel!). Otherwise, it may happen that you get a partial message on the receiver end (e.g. attachments already there, but transaction file still missing).

2.6 Remote Administration Tools

If you have a distributed IBM WebSphere MQ system, it is quite hard to do administration of all IBM WebSphere MQ server on the corresponding local machine. Instead, there are some remote maintenance tools available; for example

- The (free-of charge) IBM Support Pac MO71 (see 8.2 “Further Sources of Information”)
- Several third-party products; search the WWW for details and current releases!

2.7 Some TC/LINK-MQ Concept Remarks

For optimum operability, TC/LINK-MQ uses only techniques that are available on all IBM WebSphere MQ platforms. This means in detail

- Triggering (active notification on message arrival) is not used. All transactions are performed by polling the queues instead.
- Fragmenting/reassembling of messages is not used. Any transaction message or attachment larger than the configured maximum message size will cause the message transmission to fail.
- Grouping functionality is not used. Attachments are transferred separately, referenced by the CorrelId given in the corresponding transaction message.
- Syncpoint feature (see 3.6.2 “Syncpoint Control”) is only used in simplest form (no two-phase commit).
- No Exits are used to keep configuration as simple as possible. However, exits can be used by 3rd party developments (see 7 “Restrictions”)

- MQ Distribution lists are not supported (nevertheless, you can have multiple active recipients in a single message to KCS)
- Publish/Subscribe features are not required.

3. TC/LINK-MQ Functionality

3.1 Overview

As IBM WebSphere MQ is supported on a wide variety of platforms, TC/LINK-MQ can be used to connect all of them to KCS. This opens the powerful messaging capabilities of KCS to all of those platforms.

To connect to TC/LINK-MQ, it is necessary to provide all messages in either XML Message Format or in the TC Open Message Format (TOM).

See the documents “TC/XML Technical Manual” and “TOM Technical Manual” for details.

The customer can make use of any IBM WebSphere MQ-supported programming language on the actual platform to interface to TC/LINK-MQ.

Examples are (see IBM documentation for a complete list!)

- COBOL, C, Assembler, and PL/I on MVS/ESA
- C, C++, and COBOL on AIX
- COBOL, PL/I, C, and C++ on Windows

The open, well-documented interface makes it easy to connect existing applications. As an additional help for easy implementation, KCS provides some sample programs in ANSI C for connecting to TC/LINK-MQ (See 8 “Appendix”).

TC/LINK-MQ makes use of the built-in character conversions provided by IBM WebSphere MQ: You can use any character code-page on any platform; IBM WebSphere MQ and TC/LINK-MQ will convert it as good as possible to/from the current TCOSS code page.

3.1.1 Basic Terms

To avoid confusion with the functionality description, there are some basic terms:

- “KCS message” stands for any message or notification, sent from an originator to one (or more) recipients.
- “MQ Message”: indicates a message on the WebSphere MQ queue. Depending on the KCS message content, any KCS message is represented by one or more MQ message(s).

There are three forms of MQ Messages involved:

- “Transaction Message”: holds all relevant message information, like message type (message/notification/dirsync), originator/recipient addresses, send options, subject, and the message text.
- “Attachment”: Any binary attachments embedded in the KCS message (optional). Attachments are referenced by a special entry in the transaction message (the “:ATT:” sequence) that indicated the CorrelId of the MQ Message holding the attachment. Attachments are put to/read from the IBM WebSphere MQ queue as a separate MQ message!
- “Report”: If anything goes wrong during IBM WebSphere MQ delivery (e.g. message expired), then Reports are generated by IBM WebSphere MQ. The application must be prepared to handle them correctly.

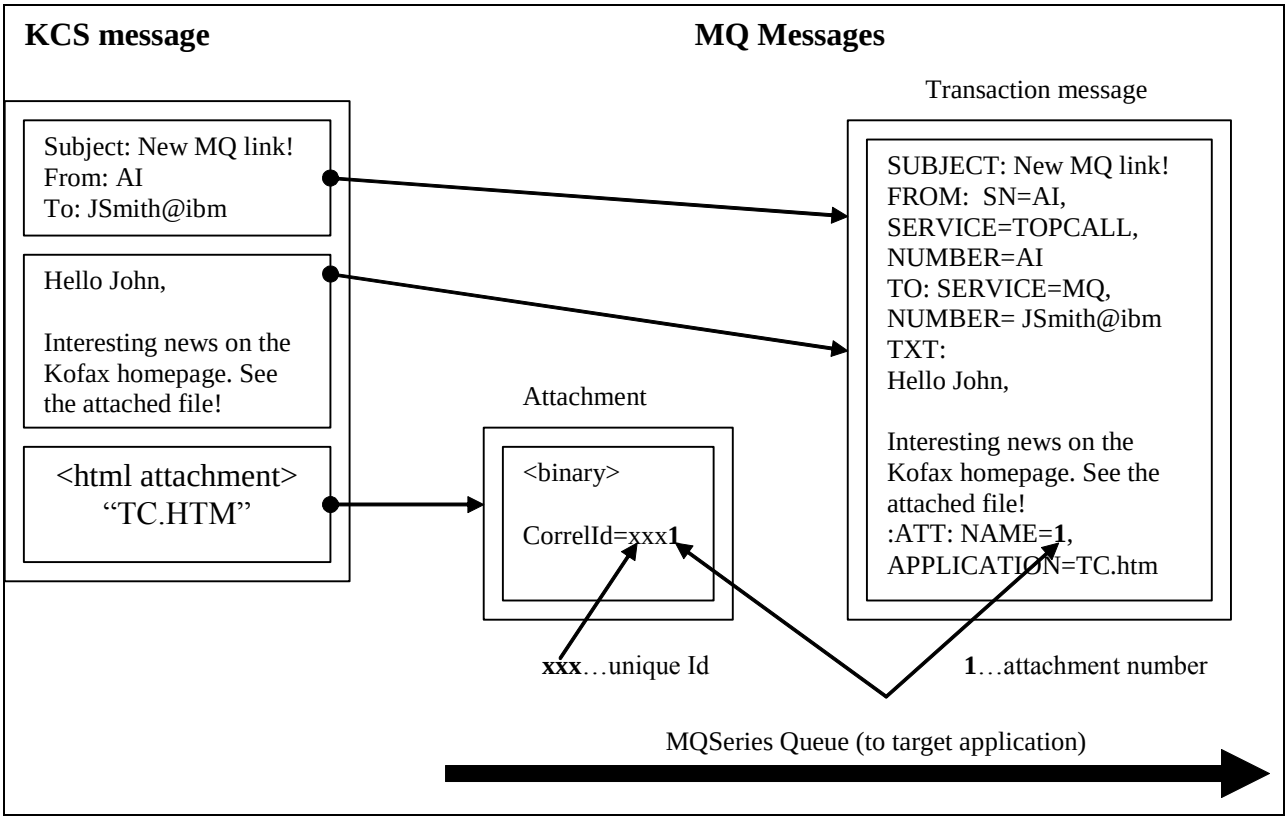
The most important properties of an MQ message are the following:

- MsgId: The unique message identifier is automatically generated by IBM WebSphere MQ. This is important to provide a defined delivery sequence (first in - first out).
- CorrelId: This is a binary field (24 byte long), used for identification of IBM WebSphere MQ messages. It is returned in any failure reports.
- CCSID: Indicates the character set of the IBM WebSphere MQ message. This is required for appropriate conversion to the destination character sets.

For playing around with TC/LINK-MQ, you find some sample programs delivered with TC/LINK-MQ that put files to IBM WebSphere MQ messages, and vice versa. See 5.8 “KCS Test Tools (MQ2File, File2MQ)” for details!

3.1.2 Basic Message Design (TOM)

Graphically, the types of messages look as follows:



- A single KCS message consists of the header (originator, recipient, subject, and some send options), the text part(s), and binary attachment(s).
- To put these KCS message onto IBM WebSphere MQ, all binary attachments are put to WebSphere MQ as separate "Attachment messages" first, with a special correlation identifier (correlation identifier is a message property of any IBM WebSphere MQ message; see 3.1.3 "Details on the CorrelId Handling")
- Then, the header and all text parts are put to a "Transaction message" in the TC Open Message Format.
- All "Attachment messages" are referenced by the Transaction message.
- The recipient on the other end of the queue must look for arriving transaction messages, and then re-insert the attachments referenced.

3.1.3 Details on the CorrelId Handling

TC/LINK-MQ relies on proper CorrelId (a IBM WebSphere MQ message property) handling to provide binary message attachments. Any connected application needs to keep the following steps:

- For any new message to send to KCS, determine a 20-byte unique binary identifier (e.g. by combining a timestamp, application identifier, sequential numbers and random numbers).
- This 20-byte unique number must be inserted to the CorrelIds of the transaction message, and all attachment messages belonging to the transaction message.
- For any attachment, you must concatenate a non-zero, sequential attachment number (4 bytes; Byte 23 = Least significant byte) to the unique binary identifier. The resulting 24 bytes are put to the attachment message CorrelId.
- The attachment number generated must also be inserted (as text!) to the transaction message: In the attachment section (:ATT:), the NAME= parameter must be filled by this number. Do not enter leading zeroes to the NAME= parameter; see the above example!

- As soon as all attachment messages are put to the queue, concatenate four zero-bytes (0x00) to the 20-byte unique binary identifier, and put it to the transaction messages CorrelId.
- A CorrelId ending with four zeroes indicates a transaction message!

When TC/LINK-MQ sends a KCS message to MQ, it builds the CorrelId according to the following logic:

CorrelId Bytes	Filled with …	Remarks
0 … 3	Seconds elapsed since 1.1.1970	Windows time() function
4 …15	TCOSS Filename	Required for notification issues
16	Number of active recipients in this KCS message (1)	Required for notification issues
17	Link number	From configuration (registry)
18, 19	KCS message sequence number	Starting from zero at Link startup; increased for every KCS message
20 .. 23	Zero for transaction message, 1 … xx for attachment messages	Byte 23 = LSB

(1) This does not limit the maximum number of active recipients in a message. If there are more than 255 active recipients in a message from KCS, then the message is split up to multiple messages.

3.2 Sending TOM Messages from WebSphere MQ to KCS

Basically, sending a message to KCS is quite easy; especially if you have a text-only message.

3.2.1 Sending Simple Text Messages

For example, you want to send a text message from your MQ client that goes out to the FAX number “6613321” on KCS. How to proceed?

First, prepare a message in the TC Open Message Format on your MQ client (or server). This may look like

```
SUBJECT=”This is my first MQ Message to KCS”
FROM: SERVICE=MQ, NUMBER=Klaus Aichner
TO: SERVICE=FAX, NUMBER=6613321
TXT:
Hello world!
```

Next, build a unique 20-byte binary identifier, paste it with four zero-bytes, and put it to the IBM WebSphere MQ message CorrelId.

Then, connect the client (or server) to the queue that will forward the message to KCS (“2TC.QUEUE” in default configuration).

Put the message to the open queue. TC/LINK-MQ will do the rest.

Attention:

- Declare the message as “persistent” to avoid loss of data in case of power failure!
- Define the queue default persistence to be “persistent” (DEFPSIST(YES))
- To enable automatic charset conversion, you need to set the “Format” to <MQFMT_STRING>.
- Avoid frequent connects/disconnects (MQCONN/MQDISC) for best performance.

The message you just created is called “TRANSACTION MESSAGE” as it starts a transaction on KCS: The sending to the recipient fax number.

Opposite to Transaction messages, there are the ATTACHMENTs. By themselves they do not start any action on KCS. Let’s look at the next chapter to see how attachments are handled.

3.2.2 Sending Messages with Attachments

The text message went through … well, fine. But you want to send your friend the latest Kofax share prices, and they are only available as spreadsheet? No problem!

Again, connect to the queue “2TC.QUEUE”.

Load the EXCEL sheet to memory, and put the binary data to a first MQ message. Set the CorrelId to a unique value NOT ending with a binary zero (Byte 20…23 of CorrelId); keep the CorrelId value for later reference (in this example, we call it "1" = hex 0x00000001)!

Put this MQ message to the queue "2TC.QUEUE".

Prepare the Transaction Message including the attachment reference:

```
SUBJECT="New share prices (EXCEL sheet)"
FROM: SERVICE=MQ, NUMBER=Klaus Aichner
TO: SERVICE=FAX, NUMBER=6613321
TXT:
Hello George,

Attached you find the current share prices of Kofax.
:ATT: NAME=1, APPLICATION=shares.xls
```

Do not forget to set the MQ message CorrelId to a value ending with zero (Byte 20…23 of CorrelId must be 0x00; Byte 0…19 must be same as for the attachment message)

Finally, put the Transaction Message to the queue "2TC.QUEUE".

Attention:

- Declare both, the attachment and transaction messages as "persistent" to avoid loss of data in case of power failure!
- Make sure to declare the attachment message(s) as binary data (MQFMT_NONE) to avoid any charset conversions!
- Both, the Transaction Message and the attachment must be given the same MQ-level priority to keep the delivery sequence!

In the Appendix of this document, you find sample source code for writing a C application that sends messages to KCS (8.7.2 "MQ2TC.C - A Simple Client for Sending to KCS (ANSI C)").

3.3 Sending TOM Messages from KCS to WebSphere MQ

In this chapter, we will do it the other way round: Send a message from KCS, and receive it on the MQ client (or server).

3.3.1 Simple Text Messages from KCS

If you start the TCfW client, you may simply address to "MQ,Klaus Aichner", type subject and any text, and press the "SEND" button … some seconds later, you will be able to receive this message in your MQ client (if your configuration is ok, of course ☺).

Now, how does the MQ Application retrieve this KCS message from the WebSphere MQ queue? There are some steps to follow:

- First of all: connect to the queue "TC2.QUEUE", and open it (for browsing first)
- Browse the queue until you find a MQ message that has a CorrelId ending with four zero-bytes. You got it – this indicates that this is a Transaction message.
- Get the message with active syncpoint option (MQGMO_SYNCPOINT; this guarantees that the message is fetched only once, and that it is invisible to other applications polling the queue at the same time). What you get is a TC Open Message Format message that may look like

```
SUBJECT="This is the first message from TCfW to MQ"
FROM: SERVICE=TOPCALL, NUMBER=AI
TO: SERVICE=MQ, NUMBER=Klaus Aichner
TXT:
Hello Klaus!

Congratulations - if you can read that, your client is working!
```

(Usually, there are much more send options in such messages; but they are not shown here for demonstration purposes. In general, you may ignore any options that your client does not support. See the "TC Open Message Format" manual for details).

- As soon as you successfully processed the Transaction Message (e.g. stored to disk, forwarded to connected application, or whatever), you may commit the “unit of work” (in this simple example, only the single transaction message). This will remove the message from the queue.
- The syncpoint control is used to make sure that nothing can be lost in case of power failure.

At this point you may wonder about the character set in which the message will arrive. Basically, this will always be the CCSID configured at the queue manager; except if you specify a different one in your MQGET message descriptor. So, you usually always get the native character set of your machine; TC/LINK-MQ and the IBM WebSphere MQ environment will convert all characters as good as possible.

Attention:

If you use queues by more than one application simultaneously, make sure to specify the MQOO_INPUT_SHARED option in the IBM WebSphere MQ “MQOPEN” call (see also the example programs in 8.7 “Samples for IBM WebSphere MQ Programming”)!

To get your messages in the expected CCSID, specify the “MQGMO_CONVERT” option!

As the heading of this chapter was “Simple Text Messages from ”, you already expected something that is not that “simple”. See next section.

3.3.2 Messages with Attachments from KCS

To receive for example incoming faxes, it is necessary that binary files are attached to a transaction message. The handling is the same up to the point where the transaction message is opened from the queue (you remember: browse for a message with CorrelId ending with four zero-bytes, then get it from the queue…). But, this time the message will look like

```
SUBJECT=”This is a message with attachment to MQ”
FROM: SERVICE=TOPCALL, NUMBER=AI
TO: SERVICE=MQ, NUMBER=Klaus Aichner
TXT:
Now I try a single attachment …

:ATT: NAME=1, APPLICATION=Image.tif
```

(Usually, there are much more send options in such messages; but they are not shown here for demonstration purposes. In general, you may ignore any options that your client does not support. See the “TC Open Message Format Manual” for details).

What you need to do now:

- Build a CorrelId from the first 20 bytes of the transaction message’s CorrelId, plus four byte from the “NAME=” parameter (Byte 23 = Least-significant).
- Get the MQ Message with that specific CorrelId (again with syncpoint control). This is the binary attachment referenced by the Transaction message!
- The real filename of the binary attachment is given by the parameter “APPLICATION”.
- There may be some more attachment parameters (a comment field, or an attachment size identifier; see the TC Open Message Format Manual for details!
- As soon as you successfully processed the Transaction Message and the Attachment (e.g. stored to disk, forwarded to connected application, or whatever), you must commit the unit of work. This will remove both from the queue (similar to the simple text message example).

If this sounds very theoretical to you (if you’d like to see a sample application): In the Appendix of this document, you find sample source code for a C application that receives messages from KCS (8.7.3 “TC2MQ.C - A Simple Client for Reception from KCS (ANSI C)”).

3.4 TC/XML Message Formats

As alternative to the TC Open Message Format (TOM) that was used up until now to put and retrieve KCS messages via TC/LINK-FI and TC/LINK-MQ, there is now also the possibility to use the KCS XML message format TC/XML. Using the TC/XML format with FI or MQ is called TC/LINK-XML.

3.4.1 TC/XML Formats

There are two XML formats used by the Link:

- TC/XML is a mapping of the classic TOM structure to XML. This is the format that is recommended to use for the costumer.
- TCXL (TC XML Link format) is the mapping of the TC/LINK internal TCSI structure to XML. This XML format can be transformed directly to the TC/LINK internal used object structure.

The conversion between these two formats is done by XSLT transformation style sheets.

Simple message in TC/XML, the KCS XML message format

```
<?xml version="1.0" encoding="UTF-8"?>
<MESSAGE xmlns="http://www.topcall.com/XMLSchema/2002/tc/xml">
   <SUBJECT>Simple example</SUBJECT>
      <TO>
      <SERVICE>FAX</SERVICE>
      <NUMBER>12345</NUMBER>
   </TO>
   <TXT>
this is a test message from TC/link-XML
   </TXT>
</MESSAGE>
```

Simple message in TCXL, the internal TC/LINK XML format

```
<?xml version="1.0" encoding="UTF-8"?>
<set_entry_ms_mail xmlns="http://www.topcall.com/XMLSchema/2002/tcxl">
   <int_msg_type>49</int_msg_type>
   <un_content.l_env_cont>
      <set_header>
         <ts_ref>minimum example</ts_ref>
         <l_recipients>
            <set_entry_rs>
               <int_del_type>1</int_del_type>
               <l_full_addr>
                  <set_full_address>
                     <ts_service>FAX</ts_service>
                     <un_public_address.set_free_address>
                        <ts_free_addr>12345##</ts_free_addr>
                     </un_public_address.set_free_address>
                  </set_full_address>
               </l_full_addr>
            </set_entry_rs>
         </l_recipients>
      </set_header>
      <obj_body_part/>
      <set_att_obj>
         <int_content_type>1076</int_content_type>
         <un_content.blk_binary.tctext>
this is a test message from TC/link-XML
   </un_content.blk_binary.tctext>
      </set_att_obj>
   </un_content.l_env_cont>
</set_entry_ms_mail>
```

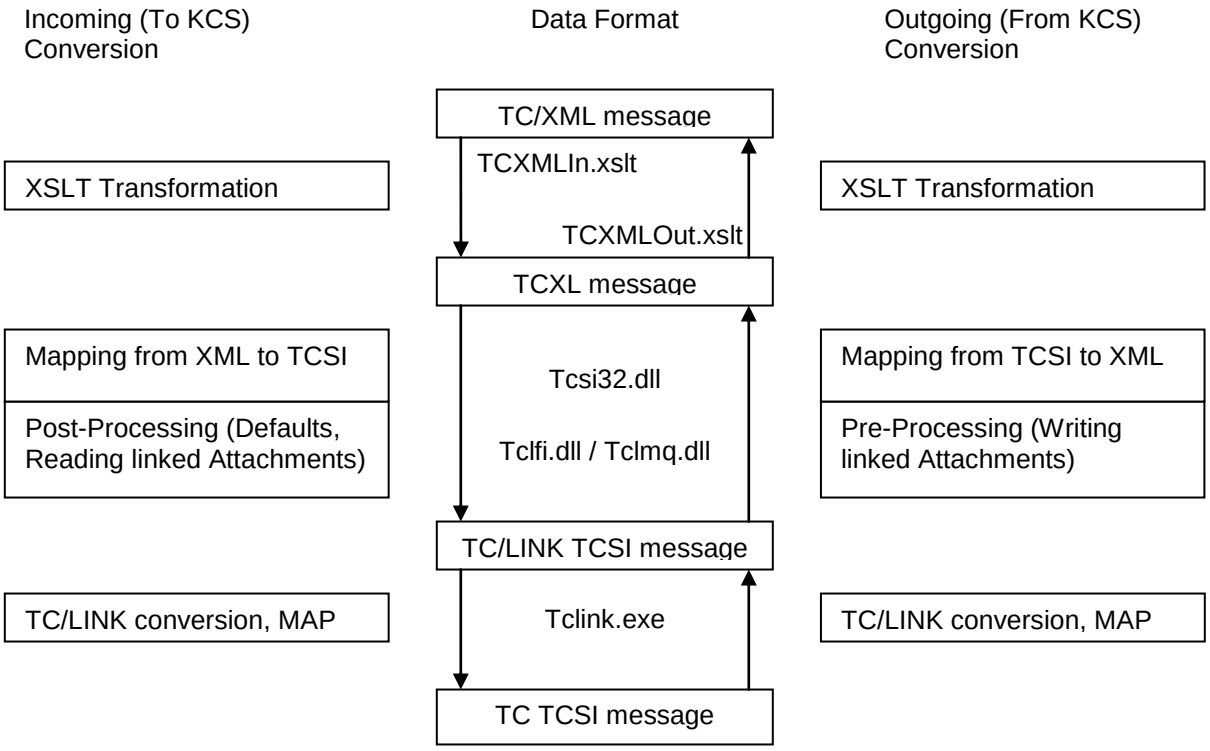
See the TC/XML manual for details on the KCS XML formats.

The indention of the XML message is only for readability, it is neither necessary for an XML message nor is it guaranteed that applications write it like that. To view a badly formatted XML file you can use for example the Internet Explorer, it does indent XML files no matter how they are written.

3.4.2 Message Conversion Flow

The following graphic shows the message conversion flow. An incoming TC/XML message is converted to TCXL by the XSLT transformation. This is mapped to the TCSI object, some defaults are added and linked

attachments are included. After that it is handed over to the general TC/Link processing. An outgoing message is handled accordingly.



The XSLT transformation style sheets can be customized to support any XML format defined by the customer. It is also possible to skip any KCS side XSLT transformation.

3.4.3 Attachments

Attachments can be embedded in the XML file, or linked similar to the classic TOM format. If the <BINARY> or <un_content.blk_binary> tag is present, the attachment is regarded to be embedded.

For outgoing messages this can be configured. The default is that attachments are just referenced and written separately.

Linked (referenced) attachment object in TC/XML.

```
<ATT>
   <NAME>1</NAME>
   <APPLICATION>report_2002.doc</APPLICATION>
   <COMMENT>some comment</COMMENT>
</ATT>
```

The <NAME> tag has the reference to find the attachment in the IBM WebSphere MQ queue (the last 4 bytes of the CorrelId). The <APPLICATION> tag defines the logical name of the attachment and is used later by the Document Converter and further on to name the attachment.

Linked (referenced) attachment object in TCXL.

```
<set_att_obj>
   <ts_comment></ts_comment>
   <ts_appl_id>report_2002.doc</ts_appl_id>
   <ts_long_file_name>report_2002.doc</ts_long_file_name>
   <ts_file_name>report_2.doc</ts_file_name>
   <int_content_type>1024</int_content_type>
   <ts_tos_folder>1</ts_tos_folder>
</set_att_obj>
```

Here the <ts_tos_folder> tag has the reference to the IBM WebSphere MQ CorrelId. <ts_appl_id> defines the logical file name, <ts_file_name> and <ts_long_file_name> define how the file should be named further on; these parameters are set to the <ts_appl_id> value if missing.

The same referenced attachment in classic TOM format

```
:ATT: NAME=1, APPLICATION=report_2002.doc
```

Embedded attachment object in TC/XML

```
<ATT>
   <NAME>1</NAME>
   <APPLICATION>testfi_A.exe</APPLICATION>
   <COMMENT></COMMENT>
   <BINARY>NpFkAMAAAAABAAAA//PAAgLAAAAAAAAAABAAAAAAAAAAAAAAAAAAAAAAAAA
   AAAAADAAA4wH66AA0mQzhgbAM1cIUhWazBCcy92ZyFWbgMWYu52b0BiYlBic15GIp5GI
   kVmLN0gCkAAAAAAAAAQUHnn2VY6FJWhpXkYFmeRiWqbGJShpXkIf56RiRY6FJqzhakIF
   oVhpXkIAAAAAAAAAAAAAAAAAAAAAAQVEAAwUADAAoUACOAAAAAAAAAAA4A8QALEgBAAAQ
   AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA=</BINARY>
</ATT>
```

The tag <BINARY> includes the binary content of the attachment encoded to Base64 characters as defined in RFC 2045 (see http://www.faqs.org/rfcs/rfc2045.html). The <NAME> tag defines the IBM WebSphere MQ CorrelId; the logical file name is taken from the <APPLICATION> tag.

Embedded attachment object in TCXL

```
<set_att_obj>
   <ts_comment></ts_comment>
   <ts_appl_id>wm_doc.fsy</ts_appl_id>
   <ts_long_file_name>wm_doc.fsy</ts_long_file_name>
   <ts_file_name>wm_doc.fsy</ts_file_name>
   <int_content_type>1024</int_content_type>
   <un_content.blk_binary>
aYUasV2U552YgAlcvZWasVGI2Vmczl2buBSNeQkOcBlcvpWZjRHXslmbrx1dt9FapNHdv
jNgKuoiGspDXBJ3Yolmdlx1cyNGXMlmbrx1dtxFZvNWAAAAANXTW90cNZ1z//HAAMAwQU
vJXYnVGAAAAABAAAAAAAAAAAAAAAAA=</un_content.blk_binary>
</set_att_obj>
```

Here the tag for the binary content is <un_content.blk_binary>. The <ts_tos_folder> tag is not necessary.

3.5 Connecting Existing Applications to TC/LINK-MQ

To connect existing MQ-enabled applications to TC/LINK-MQ, you have two basic possibilities:

- Write an interface that converts the application data to the TC/XML or TC Open Message Format BEFORE putting it to MQ.
- Put the data to MQ “as is”, and write a so-called “WebSphere MQ Message Conversion Exit”.

Just a few words on the second possibility (as the first one is 100% application-specific – no need to say something about it):

- By means of IBM WebSphere MQ message conversion exits, you can change the content of IBM WebSphere MQ messages on the way to their destination.
- IBM WebSphere MQ message conversion exits are always assigned to IBM WebSphere MQ channels (this implies that you have at least one channel involved if you want to use exits).
- Message exits can be configured on both ends of a server-to-server communication channel (on the IBM WebSphere MQ server that sends the message as well as on the one that receives the message).
- The message exit is called on both, sending and receiving IBM WebSphere MQ messages via the IBM WebSphere MQ channel.
- You can download some illustrative samples from the IBM WebSphere MQ homepage; the basic description of message exits is in the “IBM WebSphere MQ Application Programming Guide”.

Note: Using IBM WebSphere MQ Exits on the KCS machines (TCOSS server, TC/LINK server) is allowed, but not supported. This means: In case of any troubles, you must remove the exits!

3.6 Error Condition Handling

The examples above show the behavior as long as anything works fine. To avoid message loss in any case of failure, you need to consider a few things.

3.6.1 Event-Log

The following Windows event-log entries are generated by TC/LINK-MQ:

Code	Severity	Description	Insertion strings
24500	Error	Queue Manager %1 not accessible (%2)	%1 … Qmgr name %2 … MQ Error code
24501	Error	Invalid Queue Manager property: %1= %2	%1 … Property name %2 … Property value
24502	Error	Queue %1 not accessible (%2)	%1 … Queue name %2 … MQ Error code
24503	Error	Invalid Queue property: %1= %2	%1 … Property name %2 … Property value
24504	Warning	Property %1 of %2 may result in %3	%1 … Property name %2 … Object name %3 … Possible effects
24505	Error	TC/LINK-MQ out of resources	-
24506	Error	Failed to create mail dependencies (%1)	%1 … MQ Error code
24507	Error	Queue %1 is full	%1 … Queue name

3.6.2 Syncpoint Control

TC/LINK-MQ supports the so-called “Single-phase commit” mechanism of IBM WebSphere MQ.

KCS messages are always put to the local queue with syncpoint control. This means: The whole “unit of work” (which is the Transaction messages plus all attachments) will only be committed, if all parts are put to the queue successfully.

If there is anything wrong during message processing (e.g. it turns out that an attachment is too large for the queue), the whole KCS message will be backed out. This makes sure that there should never be any lost attachments hanging around on the queue.

Another example illustrating the benefits of syncpoint control is the case of power failure: if power goes down just the moment the application puts attachment messages, but the transaction message is not yet put to the queue, then the whole unit of work will automatically be backed out (as it is not yet committed).

On any case of backout, the application must issue retries; after repeated failure (by default, backout count more than 3), it shall either send a non-delivery notification to the message originator, or a problem report to the operator.

Both, TC/LINK-MQ and the connected application have to follow these rules.

Two-phase commit is a more advanced transaction handshake protocol. As there are some platforms / configurations of IBM WebSphere MQ that do not support two-phase commit (e.g. IBM WebSphere MQ server on OS/400; all IBM WebSphere MQ clients), TC/LINK-MQ does not make use of it.

However, in short, a basic explanation of the mechanism:

- While single-phase commit only involves the queue and the application putting / getting messages to / from it, two-phase commit can involve the final destination of the message. Example: if the message put to IBM WebSphere MQ is destined for a DB-2 database; the message will be successfully committed as soon as it is successfully stored at the database!
- Before any application does a put/get on a queue, it checks if all involved applications (“resource managers”) are ready for action (MQBEGIN). This is done via a so-called “transaction manager” that coordinates all involved applications.
- As soon as all applications responded “OK”, the requesting application performs its actions (get/put).
- As soon as the requesting application has finished its “unit of work”, it asks all involved applications to either COMMIT (on success) or BACKOUT (on failure) any actions performed within the unit of work.

- Depending on the status returned from all involved applications, the transaction manager issues the final COMMIT or BACKOUT to all involved applications.

3.6.3 Notification Handling

In messaging, you can never assure 100% delivery. This is due to the fact that e.g. recipient addresses may be wrong, fax numbers have been changed, or whatever. But, as soon as the message is put to the queue, and the unit of work is committed, you have no chance to back it out again …

So, you have two main possibilities:

- Send all undeliverable messages to a configured operator
- Send a non-delivery notification (delivery failure report) to the message originator

TC/LINK-MQ basically supports both possibilities; however, the non-delivery notification is usually the preferred one, as it requires no operator interaction.

If requested, TC/LINK-MQ will build notifications on delivery/non-delivery of messages to KCS.

TC/LINK-MQ can be configured to request notifications from connected applications (the “at next node” configuration in the advanced setup; described in the TCLINK manual). Connected applications must build notifications if they are requested; see the “TC Open Message Format” manual for notification format details!

A special configuration of TC/LINK-MQ assumes messages as “successfully delivered” as soon as they are put to the local queue: to get that behavior, change the “at next node” configuration to “NO” (or, in the Windows registry, set “…\General\NotifMail” to 0). Then, the application does not need to build any notifications (but must ensure delivery, with fallback to an operator if the primary destination fails!).

Note: To avoid notification loops (non-delivery for non-delivery …), you should never send a notification in response to a notification!

3.6.4 IBM WebSphere MQ Reports (KCS to MQ)

If there is something wrong on the way to the destination (e.g. queue managers are down, MQ messages expired because of this), then IBM WebSphere MQ builds so-called “Report Messages” if requested. TC/LINK-MQ takes these Report Messages to update the sending status of the KCS message (depending on the Report reason, either issues retries, or send a non-delivery to the KCS message originator).

List of all Report reasons:

Report Reason code	TCfW “Response” text	Resulting break code
MQFB_QUIT	"MQ shutdown request"	ST_BREAK2
MQFB_EXPIRATION	"Message Expired"	ST_END_OF_RETRIES
MQFB_COA	"Msg arrived (COA)"	ST_DELIVERED
MQFB_COD	"Msg delivered (COD)"	ST_DELIVERED
MQFB_PAN	"Action ok (PAN)"	ST_DELIVERED
MQFB_NAN	"Action failed (NAN)"	ST_END_OF_RETRIES
MQFB_CHANNEL_COMPLETED	"Channel completed"	ST_SENT
MQFB_CHANNEL_FAIL_RETRY	"Channel failed-retry"	ST_BREAK2
MQFB_CHANNEL_FAIL	"Channel failed"	ST_BREAK2
MQFB_APPL_CANNOT_BE_STARTED	"Appl. start error"	ST_BREAK2
MQFB_TM_ERROR	"transmission error"	ST_BREAK2
MQFB_APPL_TYPE_ERROR	"Appl type error"	ST_END_OF_RETRIES
MQFB_STOPPED_BY_MSG_EXIT	"Stopped by Msg exit"	ST_END_OF_RETRIES
MQFB_XMIT_Q_MSG_ERROR	"Xmit queue error"	ST_BREAK2
MQFB_DATA_LENGTH_ZERO	"Data length zero"	ST_END_OF_RETRIES
MQFB_DATA_LENGTH_NEGATIVE	"Data length negative"	ST_END_OF_RETRIES
MQFB_DATA_LENGTH_TOO_BIG	"Msg Too Big"	ST_END_OF_RETRIES
MQFB_BUFFER_OVERFLOW	"Buffer overflow"	ST_END_OF_RETRIES
MQFB_LENGTH_OFF_BY_ONE	"Length off by one"	ST_BREAK2
MQFB_IIH_ERROR	"IIH error"	ST_BREAK2
MQFB_NOT_AUTHORIZED_FOR_IMS	"IMS auth error"	ST_BREAK2
MQFB_IMS_ERROR	"IMS error"	ST_BREAK2
Other	“Unknown Reason %1”	ST_BREAK2

Note 1:

- See 8.3 “Glossary / Abbreviations” for the most important abbreviations.
- ST_BREAK2 means “8 retries”
- ST_DELIVERED indicates success (positive termination on KCS)
- ST_END_OF_RETRIES failure without retries (negative termination on KCS).
  (all values for default KCS configuration)

Note 2:

TC/LINK-MQ can only handle report messages if it is configured to request notifications from IBM WebSphere MQ (… the “at next node” configuration is enabled in the advanced TCLINK setup)! If this configuration is disabled, messages to MQ are terminated as soon as they are successfully put to MQ; therefore, no more status update/retries are possible. In this case, use the dead-letter queue to get indication of message delivery failures.

3.6.5 IBM WebSphere MQ Reports (MQ to KCS)

An application sending messages to KCS must be aware of two types of notifications.

- The first one is the TC Open Message Format notification as described in 8.1.1 “Notifications from ” or the corresponding manual (“TC Open Message Format”).
- The second ones are various so-called “reports” that are generated by IBM WebSphere MQ as the message delivery progresses. This depends on the report type requested from the application (any type of report can be requested – or not requested – by the application putting the message to IBM WebSphere MQ).

Example:

If a COA (“Confirmal of arrival”) report is requested, it is sent as soon as the message arrives on the destination queue manager (no TC/LINK-MQ action involved yet).

If a COD (“Confirmal of Delivery”) report is requested, this is sent as soon as the message is successfully processed and put to the KCS server (No delivery to final destination, e.g. fax machine, yet!)

Regardless of the selected IBM WebSphere MQ report level, a TOM format notification is sent (if requested) as soon as the message is either delivered successfully, or retries count goes zero (neg. notification). So, at worst case, you may get a positive report from MQ (COD ... arrived on KCS), but a nondelivery in TOM Format afterwards (e.g. “bad fax number”).

The reasons for keeping a separate TOM Format notification are mainly the following:

- IBM WebSphere MQ reports offer only a very limited set of fields for failure indication. e.g. no strings where we can tell "no fax machine detected at destination"
- IBM WebSphere MQ reports also offer only the CorrelId for matching the report to the original message. These 24 binary bytes are not very comfortable compared to the five times 128 byte text (C1...C5) given in the TOM Format notification.

3.6.6 Dead-Letter Queue

If there is a dead-letter queue configured on the local queue manager, and TC/LINK-MQ is configured to poll it (see 8.5 “Registry Values Used by TC/LINK-MQ”), all MQ messages found on this queue will be sent to the configured KCS operator. In a small English text part, TC/LINK-MQ will indicate the reason for the message to be on the dead-letter queue.

Note:

- When sending from KCS to MQ, and not requesting notifications from MQ (the “at next node” configuration in the TCLINK setup is off), then make sure to define and poll a dead-letter queue! Otherwise, you will not be notified if there are any failures after putting a message to MQ (e.g. transmission queue authorization failures, invalid destination queue names or similar).
- The “expiry” configuration of TC/LINK-MQ is ignored when “at next node” is disabled.

What would happen if “expiry” is set while “at next node” is disabled?

- A message is sent from KCS to IBM WebSphere MQ.

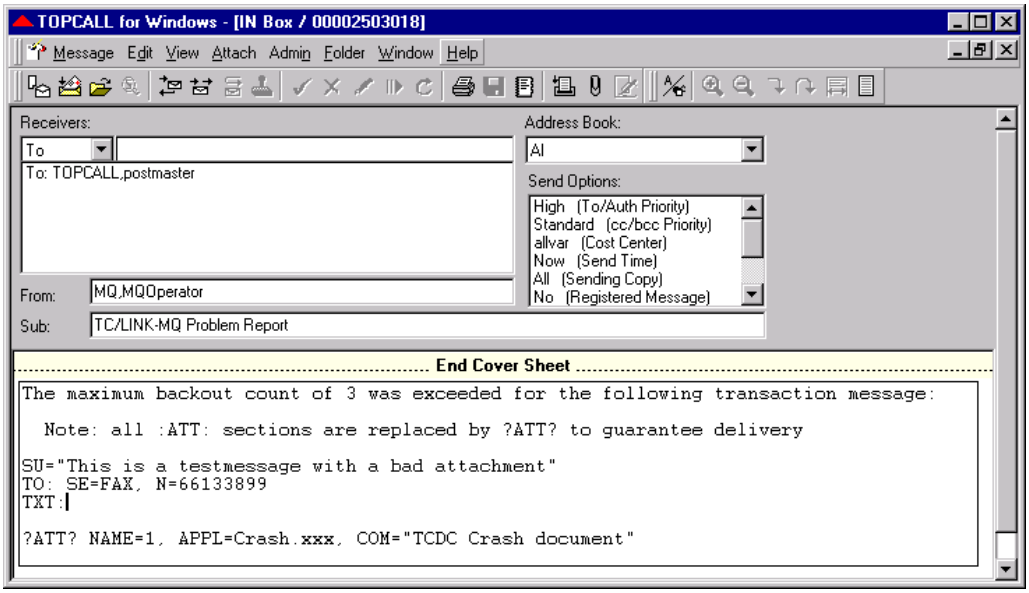
- If the “at next node” setting is disabled on TC/LINK-MQ, the message is marked “terminated” (Sent OK) on KCS as soon as it is put to IBM WebSphere MQ.
- Usually, IBM WebSphere MQ can be assumed to be a secure transport; this means, no messages can get silently lost (at worst, they end up in the dead letter queue).
- BUT: Expired IBM WebSphere MQ messages do not end in a dead-letter queue; they generate a “expired” report message instead.
- As the message is already terminated on KCS (no more status update possible), TC/LINK-MQ can not correctly handle the “expired” report.
- As a result, the message will be displayed “Sent ok” on KCS – but in fact it would be silently lost!

3.7 Backout Count Exceeded

When TC/LINK-MQ encounters any fatal error during message processing (e.g. the Document Converter crashed due to a corrupted document), the message to KCS is backed out; this means, it is re-inserted to the queue it came from.

IBM WebSphere MQ maintains a so-called “Backout Count” for messages, which is increased any time a message is backed out. If TC/LINK-MQ finds a IBM WebSphere MQ message with a backout count of more than 3 (hard-coded value), it sends the message as a problem report to the operator.

Example “Backout Count Exceeded” Report:



3.8 PUT Error Handling

When TC/LINK-MQ finds any problems to put messages on the configured TC2.QUEUE or NOTF.QUEUE, it issues retries according to the following break codes:

MQPUT Reason code	TCfW “Response” text	Resulting break code	Restart
MQRC_ADAPTER_NOT_AVAILABLE	"Adapter missing"	ST_BREAK4	TRUE
MQRC_ADAPTER_SERV_LOAD_ERROR	"Can't load adapter"	ST_BREAK4	TRUE
MQRC_API_EXIT_LOAD_ERROR	"API exit load error"	ST_BREAK4	TRUE
MQRC_ASID_MISMATCH	"ASID Mismatch"	ST_BREAK4	TRUE
MQRC_COD_NOT_VALID_FOR_XCF_Q	"invalid COD for XCF"	ST_END_OF_RETRIES	FALSE
MQRC_CICS_WAIT_FAILED	"Wait failed (CICS)"	ST_BREAK4	FALSE
MQRC_CONNECTION_BROKEN	"Connection broken"	ST_BREAK2	TRUE
MQRC_CONNECTION_NOT_AUTHORIZED	"Not authorized"	ST_END_OF_RETRIES	FALSE
MQRC_CONNECTION_QUIESCING	"Connection quiescing"	ST_BREAK2	TRUE
MQRC_CONNECTION_STOPPING	"Connection stopping"	ST_BREAK2	TRUE
MQRC_CONTEXT_HANDLE_ERROR	"Context handle error"	ST_BREAK2	TRUE
MQRC_CONTEXT_NOT_AVAILABLE	"Context unavailable"	ST_BREAK2	TRUE

MQRC_DH_ERROR	"Dist. header error"	ST_END_OF_RETRIES	FALSE
MQRC_EXPIRY_ERROR	"Invalid expiry time"	ST_BREAK2	TRUE
MQRC_FEEDBACK_ERROR	"Invalid Feedback"	ST_END_OF_RETRIES	FALSE
MQRC_GROUP_ID_ERROR	"GroupId error"	ST_END_OF_RETRIES	FALSE
MQRC_HCONN_ERROR	"Invalid Conn. handle"	ST_BREAK2	TRUE
MQRC_HOBJ_ERROR	"Invalid Obj. handle"	ST_BREAK2	TRUE
MQRC_INCOMPLETE_GROUP	"Incomplete Group"	ST_END_OF_RETRIES	FALSE
MQRC_INCOMPLETE_MSG	"Incomplete Message"	ST_END_OF_RETRIES	FALSE
MQRC_INCONSISTENT_PERSISTENCE	"Persistence error"	ST_END_OF_RETRIES	FALSE
MQRC_INCONSISTENT_UOW	"UOW Inconsistent"	ST_END_OF_RETRIES	FALSE
MQRC_MD_ERROR	"Invalid MQMD"	ST_END_OF_RETRIES	FALSE
MQRC_MDE_ERROR	"Invalid MQMDE"	ST_END_OF_RETRIES	FALSE
MQRC_MSG_FLAGS_ERROR	"Invalid Msg flags"	ST_END_OF_RETRIES	FALSE
MQRC_MSG_TOO_BIG_FOR_Q	"Message too big"	ST_BREAK5	FALSE
MQRC_MULTIPLE_REASONS	"Multiple reasons"	ST_BREAK5	FALSE
MQRC_NOT_OPEN_FOR_OUTPUT	"Not open for output"	ST_BREAK2	TRUE
MQRC_NOT_OPEN_FOR_PASS_ALL	"Not open for PASS_ALL"	ST_BREAK2	TRUE
MQRC_NOT_OPEN_FOR_PASS_IDENT	"Not open for PASS_ID"	ST_BREAK2	TRUE
MQRC_NOT_OPEN_FOR_SET_ALL	"Not open for SET_ALL"	ST_BREAK2	TRUE
MQRC_NOT_OPEN_FOR_SET_IDENT	"Not open for SET_ID"	ST_BREAK2	TRUE
MQRC_OBJECT_CHANGED	"Object has changed"	ST_BREAK2	TRUE
MQRC_OBJECT_DAMAGED	"Object damaged"	ST_BREAK2	TRUE
MQRC_OFFSET_ERROR	"Invalid seg. offset"	ST_BREAK5	FALSE
MQRC_OPTIONS_ERROR	"Options error"	ST_BREAK5	TRUE
MQRC_ORIGINAL_LENGTH_ERROR	"Invalid orig. length"	ST_END_OF_RETRIES	FALSE
MQRC_PAGESET_ERROR	"Pageset error"	ST_BREAK3	TRUE
MQRC_PAGESET_FULL	"Pageset full"	ST_BREAK3	TRUE
MQRC_PERSISTENCE_ERROR	"Invalid persistence"	ST_BREAK5	TRUE
MQRC_PERSISTENT_NOT_ALLOWED	"Pers. not allowed"	ST_BREAK2	TRUE
MQRC_PMO_ERROR	"Invalid PMO"	ST_BREAK5	FALSE
MQRC_PMO_RECORD_FLAGS_ERROR	"Invalid PMO flags"	ST_BREAK5	FALSE
MQRC_PRIORITY_ERROR	"Invalid priority"	ST_BREAK5	FALSE
MQRC_PUT_INHIBITED	"PUT inhibited"	ST_BREAK3	TRUE
MQRC_PUT_MSG_RECORDS_ERROR	"Invalid Putmsg rec's"	ST_BREAK5	FALSE
MQRC_Q_DELETED	"Queue deleted"	ST_BREAK3	TRUE
MQRC_Q_FULL	"Queue full"	ST_BREAK3	FALSE
MQRC_Q_MGR_NAME_ERROR	"Qmgr name error"	ST_BREAK2	TRUE
MQRC_Q_MGR_NOT_AVAILABLE	"Qmgr not available"	ST_BREAK2	TRUE
MQRC_Q_MGR_QUIESCING	"Qmgr quiescing"	ST_BREAK3	TRUE
MQRC_Q_MGR_STOPPING	"Qmgr shutdown"	ST_BREAK3	TRUE
MQRC_Q_SPACE_NOT_AVAILABLE	"Qmgr disk full"	ST_BREAK3	TRUE
MQRC_REPORT_OPTIONS_ERROR	"Invalid report opts"	ST_BREAK5	FALSE
MQRC_RESPONSE_RECORDS_ERROR	"Invalid response recs"	ST_BREAK5	FALSE
MQRC_RESOURCE_PROBLEM	"Resource error"	ST_BREAK3	TRUE
MQRC_SEGMENT_LENGTH_ZERO	"Zero-length segment"	ST_BREAK5	FALSE
MQRC_STORAGE_CLASS_ERROR	"Invalid storage class"	ST_BREAK5	FALSE
MQRC_STORAGE_NOT_AVAILABLE	"Insufficient storage"	ST_BREAK3	TRUE
MQRC_SUPPRESSED_BY_EXIT	"Suppressed by exit"	ST_END_OF_RETRIES	FALSE
MQRC_SYNCPOINT_LIMIT_REACHED	"Syncpoint limit"	ST_END_OF_RETRIES	FALSE
MQRC_SYNCPOINT_NOT_AVAILABLE	"Syncpoint unavail."	ST_BREAK5	FALSE
MQRC_UNEXPECTED_ERROR	"Unexpected PUT error"	ST_BREAK5	TRUE
MQRC_UOW_NOT_AVAILABLE	"UOW Unavailable"	ST_BREAK5	FALSE
MQRC_WRONG_MD_VERSION	"Invalid MQMD version"	ST_BREAK5	TRUE
All other	Various	ST_BREAK2	FALSE

Note: You find some useful abbreviations in 8.3 “Glossary / Abbreviations”.

3.9 WebSphere MQ Cluster Support

TC/LINK-MQ supports Queue Manager Clusters. The IBM WebSphere MQ manual on Queue Manager Clusters describes the following:

"As well as setting up clusters to reduce system administration, you can create clusters in which more than one queue manager hosts an instance of the same queue. You can organize your cluster such that the queue managers in it are clones of each other, able to run the same applications and have local definitions of the same queues.

The advantages of using clusters in this way are:

- Increased availability of your queues and applications
- Faster throughput of messages
- More even distribution of workload in your network

Any one of the queue managers that hosts an instance of a particular queue can handle messages destined for that queue. This means that applications need not explicitly name the queue manager when sending messages. A workload management algorithm determines which queue manager should handle the message."

3.9.1 WebSphere MQ Configuration

The following behavior is described by IBM WebSphere MQ:

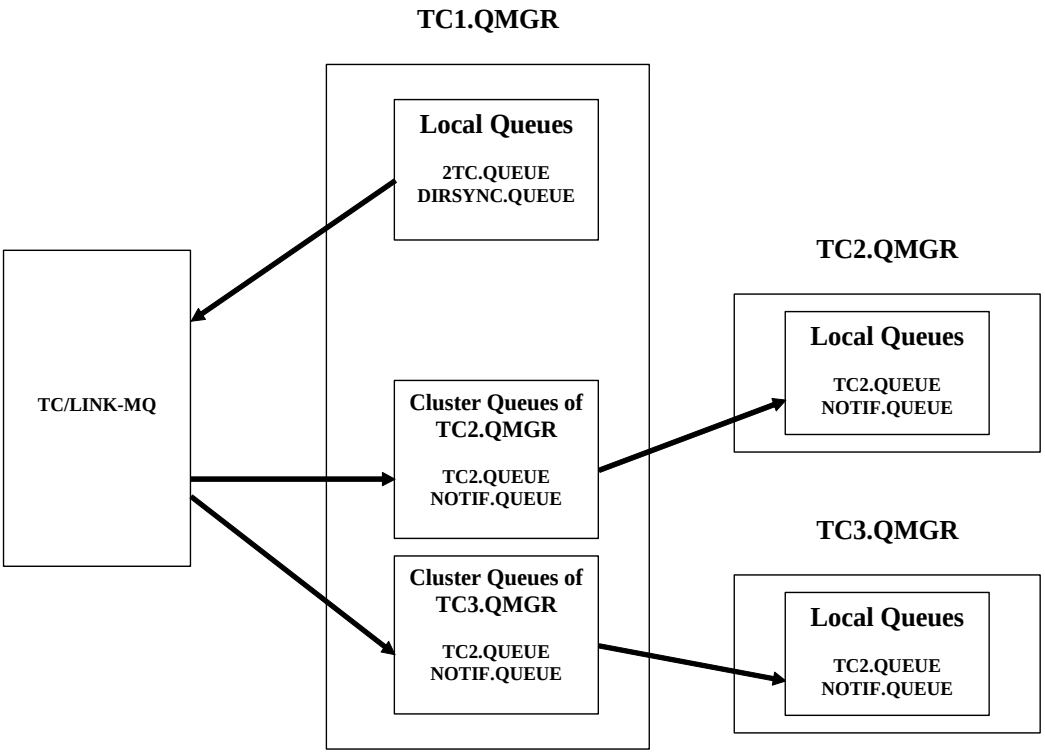
"If an application opens a target queue so that it can read messages from it or set its attributes, the MQOPEN call operates only on the local version of the queue.

If an application opens a target queue so that it can write messages to it, the MQOPEN call chooses between all available instances of the queue. Any local version of the queue is chosen in preference to other instances. This might limit the ability of your applications to exploit clustering."

For the operation of TC/LINK-MQ this means the following:

- The queue manager TC/LINK-MQ connects to has to have the read queues (that are the 2TC-Queue and the Dirsync-Queue) defined locally.
- If you connect to a queue manager, that has the TC2-Queue and the Notify-Queue defined locally, always the local queue will be used - no workload balancing will be done. So the TC2-Queue and the Notify-Queue should be Cluster Queues in order to distribute the messages between the queue managers.

The following figure shows the KCS side of a possible WebSphere MQ Cluster configuration:



TC/LINK-MQ connects to TC1.QMGR. The TC1.QMGR has to define local instances of the queues TC/LINK-MQ wants to read from (2TC.QUEUE and DIRSYNC.QUEUE). On the other hand, it is not allowed to define local instances of the write-queues (TC2.QUEUE and NOTIF.QUEUE) or else no workload balancing would be possible. Instead TC1.QMGR has the write-queues defined as cluster queues that reside on the queue managers TC2.QMGR and TC3.QMGR. Outgoing KCS messages will be put alternately either to the queues of TC2.QMGR or TC3.QMGR.

3.9.2 TC/LINK-MQ Configuration

On TC/LINK-MQ you have to enable WebSphere MQ Cluster Support explicitly. The following registry key has to be set:

HKEY_LOCAL_MACHINE\Software\TOPCALL\TCLINKMQ\Options

Registry Key	Type	Default	Description
ClusterSupport	DWORD	0	0 No WebSphere MQ Cluster Support 1 WebSphere MQ Cluster Support enabled with OpenOptions MQOO_BIND_ON_OPEN 2 WebSphere MQ Cluster Support enabled with OpenOptions MQOO_BIND_NOT_FIXED; this setting is necessary when using ALIAS queues; Restriction: only XML format is allowed with XMLAttachments = “embedded”

When no cluster support is enabled, TC/LINK-MQ opens the WebSphere MQ queues at start-up with OpenOptions MQOO_BIND_ON_OPEN. So during the runtime of TC/LINK-MQ, always the same queues will be used.

When cluster support is enabled, TC/LINK-MQ only connects to the WebSphere MQ queues it wants to read from (2TC-Queue and Dirsync-Queue). The queues TC/LINK-MQ is writing to (TC2-Queue and Notify-Queue) are only opened when there is a message to write and closed after each message. Note that a KCS message may consist of more than one WebSphere MQ messages. Such a message sequence is written to the same WebSphere MQ queue during one open – close cycle. The following KCS message will than be written to another WebSphere MQ queue if the MQ system provides an alternative queue.

3.9.3 Hints

Every IBM WebSphere MQ application has to connect to a specific queue manger. If you do not specify a queue manager in the TC/LINK-MQ configuration, the default queue manager will be used. Please refer to the WebSphere MQ Administration Manual how to configure this - it depends on the used platform.

If you connect to a queue manager, that has the TC2-Queue and the Notify-Queue defined locally, always the local queue will be used - no workload balancing will be done.

The 2TC-Queue and the Dirsync-Queue have to be defined locally on the queue manger TC/LINK-MQ connects to or else TC/LINK-MQ won’t start-up.

3.9.4 IBM WebSphere MQ Background

A queue manager is an organizational unit of IBM WebSphere MQ. All objects (e.g. queues) belong to a queue manager. Several queue managers can reside on one computer. A queue manager could also be called an instance of WebSphere MQ.

A local queue is defined on the own queue manager.

A cluster queue is defined on another queue manager, but there can be a copy (instance, clone) of the queue on each other queue manager of the cluster.

For more information, especially on how to define and configure IBM WebSphere MQ, please see the WebSphere MQ documentation.

3.10 Unicode Support

Since KCS 9.2, TC/LINK-MQ supports TOM Unicode transaction files. By default, incoming transaction files (to KCS) are requested to be in the configured PC code page, and outgoing transaction files (from

KCS) are written in PC code page as before, however both directions can also be configured to request/write the TOM transaction files in UTF-8 or UTF-16.

Additionally, it is possible for the inbound direction (to KCS) to turn conversion of transaction files off (registry Options\NoConvert=1). In this case IBM WebSphere MQ does not convert the file, TC/LINK-MQ pulls it as it is. UTF-8 or UTF-16 text files are recognized by their byte order mark (BOM), if not the file is interpreted in the configured PC code page.

TC/XML transaction files are handled by default as UTF-8 files as before.

See *Unicode Installation Guide* for general information about Unicode.

3.10.1 General Unicode Configuration

General KCS Unicode support is by default enabled. For compatibility with Exit-Dlls it can be disabled during setup by unchecking “Unicode supported”.

You can disable Unicode support already before starting setup by changing the file defaults.ini:

```
[TCLINKMQ]
Setup\ModuleSupportsUnicode=1
```

In the [TCLINKMQ] section, set ModuleSupportsUnicode=0.

After installation, you can configure Unicode support via Registry:

HKEY_LOCAL_MACHINE\SOFTWARE\TOPCALL\TCLINKMQ\General

Registry Value	Type	Default	Description
UnicodeSupported	DWORD	1	Enable general TC/LINK support of KCS Unicode

3.10.2 TOM Configuration for Transaction Files

The following registry configuration defines the coding of TOM transaction files.

HKEY_LOCAL_MACHINE\SOFTWARE\TOPCALL\TCLINKMQ\Options

Registry Value	Type	Default	Description
In\TomCodePage	STRING	“PCCodePage”	Coding as requested when pulling the transaction file from IBM Websphere MQ. Possible values are: PCCodePage Windows code page as defined by registry key General\PCCodePage Utf8Bom UTF-8 with byte order mark Utf16LeBom UTF-16 Little Endian with byte order mark
Out\TomCodePage	STRING	“PCCodePage”	Coding of outgoing TOM transaction files, as they are put to IBM Websphere MQ. Possible values: PCCodePage Windows code page as defined by registry key General\PCCodePage Utf8Bom UTF-8 with byte order mark Utf16LeBom UTF-16 Little Endian with byte order mark

In order to support Unicode you have to change the settings of these keys. Of course, the application on the other side of the IBM Websphere MQ system must also to support Unicode. Depending on this processing you might want to choose the TomCodePage settings.

On TC/LINK-MQ side the most efficient setting is “Utf16LeBom”, as this is how messages are processed internally.

4. Prerequisites for Installation of TC/LINK-MQ

See the general TCLINK manual for all common prerequisites, like minimum Windows versions, hardware requirements, or similar.

TC/LINK-MQ requires TC/SP 7.08 or higher (license handling)

The appropriate transport to the connected applications must be installed (one of TCP/IP, NetBIOS, IPX, or SNA/LU6.2). See the IBM WebSphere MQ documentation for supported protocol stacks.

In order to use TC/LINK-MQ with TC/XML on a Windows-computer, you have to install Internet Explorer 5.5 or higher.

5. Installation

Basically, there are two main parts that needs to be installed and configured:

- The TC/LINK-MQ configuration.
- The IBM WebSphere MQ server: queue manager, queues, channels

5.1 Checklist for Setup

Before you begin installation, you should gather all required environmental information.

5.1.1 KCS-Related Information

KCS Server CPU number	
KCS Link Server CPU number	
KCS Server version	
TC/LINK-MQ license	Key: Expire Date: Registrations:
Postscript license (optional)	Key: Expire Date: Registrations:
PCL5 license key (optional)	Key: Expire Date: Registrations:
GIF license key (optional)	Key: Expire Date: Registrations:
File Reporter license key (optional)	Key: Expire Date: Registrations:
KCS Server Name	
Link Type to KCS Server, transport (RPC / Native)	
Secondary Server Name (tandem servers only)	
Link Type to secondary Server (tandem servers only)	
KCS Link User Name	
KCS Link User Password	

5.1.2 Windows-Related Information

Windows TCLINK username (max. 12 characters long!)	
Windows TCLINK user password	
Windows TCLINK user domain	
Windows TCDCEXE username (foreground TCDC only)	
Windows TCDCEXE user password (foreground TCDC only)	
Windows TCDCEXE user domain (foreground TCDC only)	

5.1.3 IBM WebSphere MQ Related Information

IBM WebSphere MQ server name (if connecting to an existing environment)	
IBM WebSphere MQ queue manager maximum message size (if connecting to an existing environment)	
IBM WebSphere MQ queue manager name (if connecting to an existing environment)	
Queue name to KCS (if connecting to an existing environment)	
Queue name for messages from KCS (if connecting to an existing environment)	

Queue name for notifications from KCS (if connecting to an existing environment)	
Queue name for Dirsync messages (if connecting to an existing environment)	
Transport parameters (for TCP/IP): Channel name: TC/LINK-MQ IP address: MQ Server IP address: IBM WebSphere MQ port number (defaults to 1414):	

5.2 Licenses

Depending on the selected message data format (see section TC/LINK-MQ Easy Installation), you need an appropriate license:

- TC/XML – TC/XML license
- TOM Classic – TC/LINK-MQ license

5.3 Installing TC/LINK-MQ

5.3.1 Windows-Specific Configurations

To set up communications to a remote IBM WebSphere MQ server, you need to install the selected network protocol (e.g. the TCP/IP stack for a TCP connection).

Define the maximum message size you want to use with TC/LINK-MQ. If you are planning to use a maximum message size of more than 4Mbytes, follow the hints given in 5.5.1 “Large Message Operation”.

5.3.2 Common Installation for all TC/LINKs

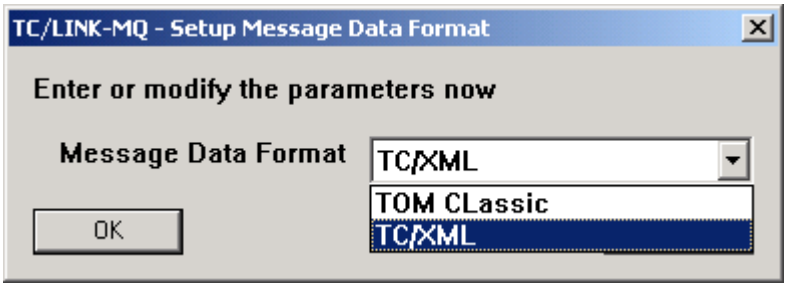
TC/LINK-MQ is one of the links in the Kofax Communication Server setup. Major part of its setup is common with all other links.

- Only for TC/SP releases lower than 7.22, you need to create a Link user for the TC/LINK-MQ on KCS (by TCfW). This is required for automatic installation of all requirements on KCS.
- Run the Setup program, fill in all general TC/LINK configuration items. Refer to the TC/LINK manual for details!
- See the following screenshots for TC/LINK-MQ – specific configuration.

Note: The configured Windows user account for TC/LINK-MQ must be at maximum 12 characters long! (IBM WebSphere MQ for Windows requirement)

5.3.3 TC/LINK-MQ Easy Installation

In the simple setup mode, you are only asked for the preferred message format:



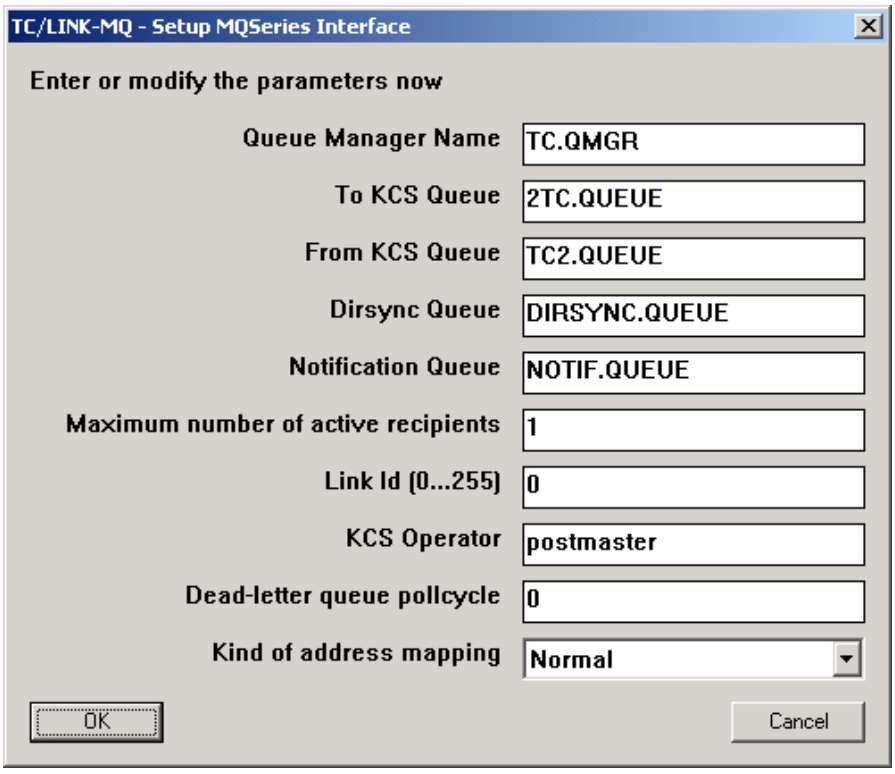
This window lets you choose between the older TOM and the newer TC/XML format.

See section 8.5 Registry Values Used by TC/LINK-MQ for all defaults used in the easy installation mode!

5.3.4 TC/LINK-MQ Advanced Installation

If you select the “Configure Advanced Features” option in the first link setup screen, the following configuration windows appear:

5.3.4.1 Setup IBM WebSphere MQ Interface



Queue Manager Name (…\Options\QueueManager): Please enter the name of the queue manager here.

Queue names (…\Options\Queue2TC, QueueNotif, QueueTC2, QueueDirsync): Please enter the name of all used queues on the queue manager. Notification queue may be the same as the “From KCS” queue, Dirsync queue may be same as “To KCS” queue.

Maximum number of active recipients (…\Options\MaxActRecp): Indicates the maximum number of active recipients in messages to MQ. If you e.g. specify “1” here, and send a message to three MQ recipients, three separate messages are sent; If you specify “5”, a single message with three active recipients will be sent. Maximum is 255.

Link Id (…\Options\LinkNum): This must be a queue-manager unique link number (is important for multiple TC/LINK-MQs working on the same set of queues).

KCS Operator (…\Options\Operator): existing KCS user to send problem reports, dead letter queue contents, or similar. The default KCS service (configured in advanced TCLINK setup) is used as service.

Dead-letter queue poll cycle (…\Options\DLQPoll): Poll dead-letter queue any <nnn> TC/LINK poll cycles. A value of zero disables the feature. If there is no Dead-letter queue defined on the configured queue manager, this feature is also disabled.

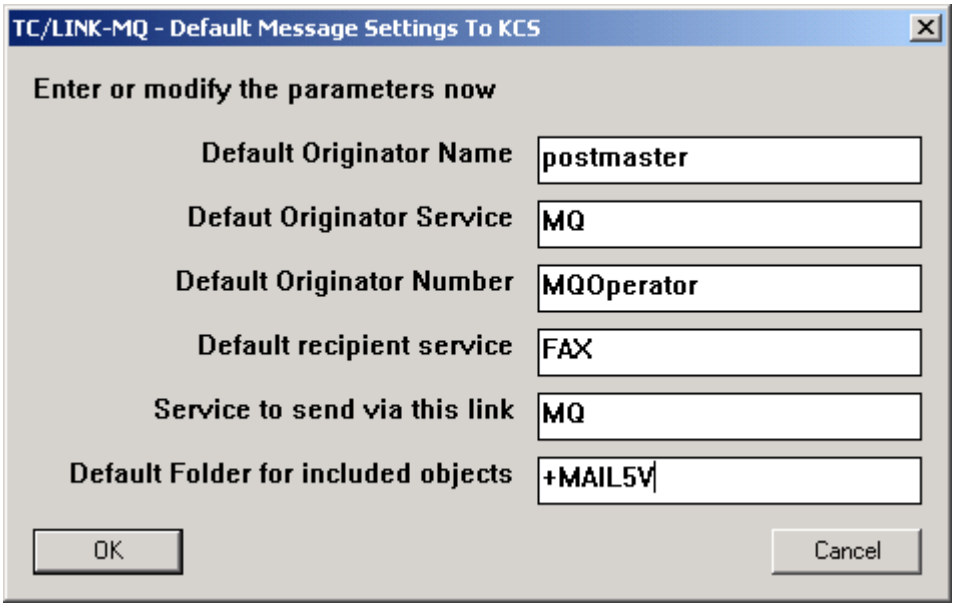
Kind of address mapping (…\Options\UseShadowUser): If you do not maintain shadow users on KCS, switch to “No shadowusers”. This setting is also important for obtaining default values: With Setting “Normal”, the template, full name etc. will be taken from the shadow user!

Note:

- All IBM WebSphere MQ names (queues, queue manager, channels etc.) are generally case-sensitive! For optimum interoperability, use UPPERCASE letters for all names!
- If you change the queue manager or queue names from the default, you need to adapt the <c:\tcoss\tclink\MQSetup.txt> file accordingly (this is not automatically changed!)

- If you are using a local IBM WebSphere MQ server, at least the 2TC.QUEUE and DIRSYNC.QUEUE must be local queues (IBM WebSphere MQ does not allow GET operations on remote queues).

5.3.4.2 Default Message Settings To KCS



Default Originator Name (…\Options\DefaultUserName): Gives the possibility to enter a KCS User-ID for the default originator. Only required if the default originator is a KCS user; if defined, it overrides the "Default originator number" configuration.

Default Originator Service (…\Options\DefaultOriginatorService): The service that shall be used if no originator is specified in the message to KCS.

Default Originator Number (…\Options\DefaultOriginatorNumber): The originator number that shall be used if none is specified in the message to KCS.

Default Recipient Service (…\Options\DefaultRecipientService): The recipient service to be used if not specified in the message to KCS.

Service to send via this link (…\Setup\ServiceMQ\Name): Indicates the Links "native" service name.

Default Folder for included objects (…\Options\DefaultFolder): This folder is used if :INC: objects are used in the message to KCS without explicit FOLDER= specification.

5.3.4.3 Other Configuration Parameters

Some special configuration items can only be accessed via Registry editor (see section 8.5 "Registry Values Used by TC/LINK-MQ" for a complete listing).

5.4 Installing the WebSphere MQ Server on the TC/LINK-MQ Computer

TC/LINK-MQ requires either a local IBM WebSphere MQ server or client. Which of them is installed, will be detected automatically at TC/LINK-MQ startup.

Reasons for using IBM WebSphere MQ server:

- When using a local IBM WebSphere MQ server, you do not need the Client Access Facility (CAF) when connecting to a server on MVS. This saves the license cost for CAF that may be quite high.
- Message Exits are only available with the IBM WebSphere MQ server.
- Automatic creation of IBM WebSphere MQ dependencies is only available with the local IBM WebSphere MQ server.

If you choose to use the local IBM WebSphere MQ server, continue with 5.4.1 "Installing the IBM WebSphere MQ Server".

Reasons for using IBM WebSphere MQ client:

- Fail-safe systems are only possible by using the IBM WebSphere MQ client (see 8.1.3 “Automatic Fail-Over” for details)
- The IBM WebSphere MQ client does not require any license (download for free from the IBM WebSphere MQ homepage – see 8.2 “Further Sources of Information”)

If you choose to use the local IBM WebSphere MQ client, continue with 5.4.2 “Installing the IBM WebSphere MQ Client”.

5.4.1 Installing the IBM WebSphere MQ Server

There are some configuration items for TC/LINK-MQ that have to be done on the IBM WebSphere MQ server.

Here we give just a short overview what needs to be done. For troubleshooting and further details, refer to the IBM documentation!

5.4.1.1 IBM WebSphere MQ Server Installation Procedure

Installation of the IBM WebSphere MQ server is easy and straight-forward. Simply put the appropriate CD into your CDROM drive, and run Setup.

Note:

- Only the English version of IBM WebSphere MQ server is supported; so, you need to select this.
- IBM WebSphere MQ requires the Windows user account it is started with to be 12 characters maximum! Therefore, the Windows user for TCLINK must be 12 characters at maximum; “Administrator” will NOT work!
- At least the 2TC.QUEUE and DIRSYNC.QUEUE must be defined as local queues (IBM WebSphere MQ does not allow GET operations on remote queues).

MAKE SURE TO APPLY CSD-04 OR LATER TO THE IBM WEBSPHERE MQ SERVER! Older releases cause memory leaks (50..100 Bytes per message) that will make continued operation impossible!

5.4.1.2 Required Configuration

If you enabled the “automatic creation of mail dependencies” at TC/LINK-MQ configuration (which is the default), then most of the configuration is done at first link startup.

What you need to define manually is the connection to the IBM WebSphere MQ environment (at least one channel in your selected network protocol). See the “IBM WebSphere MQ for Windows Quick Beginnings” manual for some hints how to do that!

All your custom entries should be entered to the < MQSetup.txt > file that is installed to the c:\tcoss\tclink directory (Especially, the connection parameters – channels, remote queues, and similar - MUST be adapted). This file is used by TC/LINK-MQ to create all dependencies.

Note: If your computer name contains a hyphen, you have to specify it with inverted comma at the channel definition, e.g.: “CONNAME('TC-MQ-SRV')”.

5.4.1.3 Automatically Performed Configuration at First TC/LINK-MQ Startup

When TC/LINK-MQ is started, and detects the presence of a local IBM WebSphere MQ server, it does the following steps (if it is configured to automatically create all mail dependencies):

- First of all, it checks whether the configured queue manager is already there and running. If yes, everything is fine; startup completed.
- If no, TC/LINK-MQ first stops the IBM WebSphere MQ service

```
"net stop IBMMqSeries"
```

- It creates a queue manager responsible for the connection to TC/LINK-MQ (if this is not already there); logfile size is set to 4MB, the secondary logfile count to 30:

```
"crtmqm -lf 1024 -ls 30 TC.QUEUE.MANAGER"
```

- Add the new queue manager to the autostart list

```
"scmmqm -a -s c:\tcoss\tclink\Mqstartup.cmd TC.QUEUE.MANAGER"
```

- Start the queue manager by starting IBM WebSphere MQ service:

```
"net start IBMMqSeries"
```

- If the startup via service fails for some reason, TC/LINK-MQ attempts a explicit start of the specified queue manager (Note: the listener <runmqlsr> required for TCP must be started seperately in this case!)

```
"strmqm TC.QUEUE.MANAGER"
```

- Start the IBM WebSphere MQ Commands tool to activate the definitions (See Appendix 8.8 for a listing of the default definition file)

```
"runmqsc TC.QUEUE.MANAGER < .\MQSetup.txt"
```

All output from these commands is written into the TC/LINK-MQ tracefile.

Attention: If you changed the queue names (via 5.3.4 "TC/LINK-MQ Advanced Installation"), you also need to edit the <C:\tcoss\tclink\MQSetup.txt> according to your changes!

5.4.1.4 What Must Be Configured for Automatic Windows Startup

With IBM WebSphere MQ for Windows, you can configure some processes that are started automatically at Windows startup. See the documentation of the "scmmqm" command (IBM WebSphere MQ System Administration Manual) for details!

Example:

```
scmmqm -a -s c:\tcoss\tclink\Mqstartup.cmd TC.QUEUE.MANAGER
```

… Will execute the given <Mqstartup.cmd> command file at Windows startup.

The file <Mqstartup.cmd> is a simple text file; to start the queue manager, a TCP listener and a server channel a remote queue manager, it may look like

```
strmqm TC.QUEUE.MANAGER
runmqlsr -t TCP -m TC.QUEUE.MANAGER
runmqchl –m TC.QUEUE.MANAGER –c 2UNIX.CHANNEL
```

Note:

- When "Create Mail dependencies" is enabled, TC/LINK-MQ will create and register a basic autostart file (for starting its own queue manager and a listener).
- Make sure to set the IBM WebSphere MQ service to startup "Automatic" (Control Panel/Services/IBM WebSphere MQ/ Startup)!
- See the "WebSphere MQ System Administration Manual" for details on automatic startup!

5.4.2 Installing the IBM WebSphere MQ Client

If you have already installed a IBM WebSphere MQ server, you can skip this step.

If you have already installed a local IBM WebSphere MQ server, but you want to use the IBM WebSphere MQ client, you need to uninstall the local IBM WebSphere MQ server first!

5.4.2.1 IBM WebSphere MQ Client Installation Procedure

Installation of the IBM WebSphere MQ client is easy and straight-forward.

- From CD: Simply put the appropriate CD into your CDROM drive and run Setup.
- Downloaded from the Internet: Unzip the downloaded file, and run Setup.

Note:

- Only the English version of IBM WebSphere MQ client is supported.
- It is recommended to apply CSD 04 (PTF W200091A) to the IBM WebSphere MQ client.

- IBM WebSphere MQ requires the Windows user account it is started with to be 12 characters maximum! Therefore, the Windows user for TCLINK must be 12 characters at maximum; “Administrator” will NOT work!

5.4.2.2 Required Configuration on the Remote IBM WebSphere MQ Server

Note that configuration of the Remote IBM WebSphere MQ Server is somewhat platform-dependent; see the platform-specific IBM documentation for details!

Basically, the following steps are required (it is also possible to use the MQ-Server Admin program for these steps, here are nonetheless the command-line parameters):

- Create a queue manager responsible for the connection to TC/LINK-MQ:

```
"crtmqm TC.QUEUE.MANAGER" (be patient, this may take a minute!)
```

- Start the queue manager:

```
"strmqm TC.QUEUE.MANAGER"
```

- Copy the TC/LINK-MQ environment definition file (“MQSetup.txt”) to the MQ server machine (See Appendix 8.8 for a listing of the default definition file!).
- Edit the “MQSetup.txt” definitions according to your needs. Especially, the connection parameters MUST be adapted!
  Note: If your computer name contains a hyphen, you have to specify it with inverted comma at the channel definition, e.g.: “CONNAME('TC-MQ-SRV')”.
- Start the IBM WebSphere MQ Commands tool to activate the definitions:

```
"runmqsc TC.QUEUE.MANAGER < MQSetup.txt"
```

- For a TCP channel to client: Start a listener (does also display some diagnostic information during operation):

```
"runmqlsr -t tcp"
```

Attention: After the listener is started, it is not possible to use that consol-window for anything else. One might get the impression, that there is no response any more, but that is ok; do not close that window!

Optional: Copy the file “AMQCLCHL.TAB” from the queue manager directory (usually located in the <qmgrs\<qmname>\@ipcc> subdirectory). We may need this file on the IBM WebSphere MQ client (… the TC/LINK-MQ server) – see 5.4.2.4.1 “Configuration Using the “AMQCLCHL.TAB” File”!

Hint: If you have to change the IP-Address of your MQ-Server, you will have no problems if you do not change the name of the computer!

Note:

- Filenames are case-sensitive on some operating systems (e.g. UNIX)!
- All IBM WebSphere MQ names (queues, queue manager, channels etc.) are generally case-sensitive! For optimum interoperability, use UPPERCASE letters for all names!

You can also make all of the MQSetup.txt settings manually.

To do so, type the following. Attention: Uses defaults for all values not explicitly specified; e.g. for maximum message size.

```
"runmqsc"
```

(starts the IBM WebSphere MQ command tool)

```
"define qlocal (2TC.QUEUE) DEFPSIST(YES)"
```

(for messages to KCS (mandatory))

```
"define qlocal (TC2.QUEUE) DEFPSIST(YES)"
```

(for messages from KCS (mandatory))

```
"define qlocal (DIRSYNC.QUEUE) DEFPSIST(YES)"
```

(for dirsync to KCS (optional))

```
"define qlocal (NOTIF.QUEUE) DEFPSIST(YES)"
```

(for notifs from KCS (optional))

```
"define channel(TC.CHANNEL) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
       DESCR(TC SERVER CHANNEL')"
"define channel(TC.CHANNEL) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
       CONNAME(193.81.166.46) QMNAME(TC.QUEUE.MANAGER) +
       DESCR('Client Connection to TC QUEUES')"
```

Check queue options ("display qlocal" in MQSC), change if necessary (IBM WebSphere MQ command tool). Most important options are

- Maximum MQ message size (for queue manager): "ALTER QMGR MAXMSGL(<bytes>)"
- Maximum MQ message size (queue): "ALTER QLOCAL(TC2.QUEUE) MAXMSGL(<bytes>)"

Close the IBM WebSphere MQ command tool by typing "<Control Z> <Enter>" or "end"

5.4.2.3 Required Rights on the Remote IBM WebSphere MQ Server

If the IBM WebSphere MQ rights are restricted on the remote IBM WebSphere MQ server (… the TC/LINK-MQ user is not a member of the "mqm" group), make sure that at least the following is satisfied:

- The TC/LINK-MQ Windows user needs a security profile on the remote server (this means e.g. for UNIX – based IBM WebSphere MQ server that there must be a UNIX user matching the Windows username)
- TC/LINK-MQ must be granted the "+connect" and "+inquire" rights for the used queue manager
- TC/LINK-MQ must be granted the "+inquire", "+browse" and "+get" rights for the 2TC.QUEUE and DIRSYNC.QUEUE
- TC/LINK-MQ must be granted the "+inquire" and "+put" rights for the TC2.QUEUE and NOTIF.QUEUE

Note:

- You can display the current rights with the <dspmqaut> command.
- To change the rights setup, use the <setmqaut> command.
- For details on these commands, the IBM WebSphere MQ Administration Manual.
- When using these commands in Windows, they require local Windows user groups as a parameter.

5.4.2.4 Required Configuration on the Local Client

As soon as you configured the remote queue manager for use with TC/LINK-MQ, you need to set up the connection parameters on the client. You have two basic choices:

5.4.2.4.1 Configuration Using the "AMQCLCHL.TAB" File

With this configuration mode, no manual connection setup is required on the IBM WebSphere MQ client side. All you need to do is to copy the file "AMQCLCHL.TAB" (uppercase is important on some platforms) from the queue manager directory on the server to the MQ Client directory (usually "c:\mqm" or "c:\mqclient"; defined either by MQCHLLIB environment variable, or in the local <mqs.ini> file, value "DefaultPrefix") of the TC/LINK-MQ server.

This file holds all required information to access the IBM WebSphere MQ server. In addition, some features are only available for this way of configuration:

- Maximum channel message size can only be configured by use of the "AMQCLCHL.TAB" file
- A special "queue manager grouping" feature can be configured (automatic fallback to secondary IBM WebSphere MQ server if the first one is down; load sharing among queue managers)

Note:

- Before copying the "AMQCLCHL.TAB" file, make sure to define both a SVRCONN and a CLNTCONN channel of same name on the server! See the "WebSphere MQ Client" manual for details.
- This file holds binary data. Therefore, do not transmit it as text (e.g. for FTP, make sure to use BINARY transmission)!
- If the IBM WebSphere MQ client for Windows does not find the <AMQCLCHL.TAB> file, it writes a message to the Windows event log indicating the path it expects to find this file.

5.4.2.4.2 Configuration Using the MQSERVER Environment Variable

If it is – for some reason – impossible to use the client configuration via the "AMQCLCHL.TAB" file, you can alternatively set up the connection to the server via a Windows environment variable (Control Panel/System/Environment/User Variables):

```
"MQSERVER" = <Communication channel>/<Communication Protocol>/<Server Address>
```

For example:

```
"MQSERVER" = "TC.CHANNEL/TCP/193.81.166.46"
```

Communication Channel: the channel name as configured on the MQ Server.

Communication Protocol: one of TCP/NetBIOS/SPX/LU62.

Server Address: the MQ Server address (in the format matching the selected Protocol – e.g.

- For TCP: "193.81.166.46" or "pcai"
- For NetBIOS: "pcai"
- For SPX: "'0a0b0c0d.804abcde23a1'
- For LU 6.2: the CPI-C side object name

Note:

- This method has the drawback that only one single MQ-Connection can be defined.
- Make sure to enter the environment variable under the system-wide environment settings. If you enter it as user variable, you will encounter problems when starting TC/LINK-MQ with TCSRV.
- In some cases, it is required to reboot the PC to make the changed environment variables active. If you find any problems that may be related to that, try rebooting the PC!
- In this configuration, maximum message size is restricted to 4MB per MQ message! There is no way to set the channel's MAXMSGL property via the MQSERVER= environment variable!

5.5 Optional: Special Configurations

5.5.1 Large Message Operation

If you are using the default IBM WebSphere MQ maximum message size of 4MB, you usually do not need to care about resources, log files or similar.

But if you want to take use of a higher limit (absolute maximum for IBM WebSphere MQ version 5 is 100MB = 104.857600 Bytes!), you need to watch for some more critical configurations:

- You need to set the queue manager maximum message size to the desired value (with the runmqsc tool, "ALTER QMGR MAXMSGL(100000000)")
- You need to set all used queue's limits to the same value ("ALTER QLOCAL(QUEUENAME) MAXMSGL(100000000)")
- If you use channels with that messages, you must set the maximum channel message size of all involved channels ("ALTER CHANNEL(CHLNAME) CHLTYPE(TYPE) MAXMSGL(100000000)")
- If you want to use this maximum message size with IBM WebSphere MQ clients, you need to provide the channel configuration in the <AMQCLCHL.TAB> file (see 5.4.2.4.1 "Configuration Using the "AMQCLCHL.TAB" File")
- Make sure that the queue manager log file size is sufficient for the desired message size. This can be calculated from the queue manager's <qm.ini> file settings (<qm.ini> is located in c:\mqm\qmgrs\<queue manager name>):

```
(LogPrimaryFiles + LogSecondaryFiles) * LogFilePages * 4kBytes
```

… must be larger than your desired maximum message size!

- Make sure that you add at least four times the maximum message size to the Windows swap file (Control Panel/System/Performance/Virtual memory/Change/Set)
- Make sure that the disk free space is sufficient to keep the enlarged swap file, plus the maximum IBM WebSphere MQ log file, plus space for temporary files (e.g. TCDC); so, there should at least be 6 times the maximum message size of free disk space!

- Make sure to set the TCLINK watchdog timeout (registry \General\AllowedDelay in minutes) to a sufficiently high value (this depends on hardware and network performance; before having test results, you may assume about one or two minute(s) per MB maximum message size).
- Make sure to disable TC/LINK-MQ trace when sending that large messages (no Maildebug, TCSIdebug, MQtrace; General trace level at maximum 0x40!)
- Make sure that your KCS server can handle the desired maximum message size (TCOSS file structure should be at least five times the maximum message size).

See also section 2.4.1.1 “Detail: Queue Manager Log Files” regarding IBM WebSphere MQ log files!

5.5.2 IBM WebSphere MQ and Firewalls

When using IBM WebSphere MQ in a distributed server environment, firewalls are a possible problem for interconnection.

There is a nice manual on that topic, available from the IBM WebSphere MQ homepage (see 8.2 “Further Sources of Information”). Look for “SupportPac MA86”.

5.5.3 IBM WebSphere MQ Security with the MQ Client

If you are not using a security exit, you can specify the MQ_USER_ID and MQ_PASSWORD environment variables in order to identify to the server.

Note:

- These environment variables must be entered to the Windows system settings. Reboot the machine is required to make them effective.
- These values are transferred in plain text over an IP connection. You may choose to use e.g. an encryption/decryption exit, or SNA with password/encryption if this is a problem to you.
- See the “IBM WebSphere MQ client” manual for further details!

5.5.4 Multiple Instances of TC/LINK-MQ on the Same Set of Queues

When you want to operate multiple TC/LINK-MQs on the same set of queues (for reasons of throughput, or fail-save system design), a few rules must be obeyed:

- All of these links must have the same basic setup (especially the same registry subkey name … leave it the default HKLM\TOPCALL\TCLINKMQ; same Link users and same “at next node” setup).
- The only thing that must be different on all of them is the link number parameter (registry …\TCLINKMQ\Options\LinkNum)! Allowed values are 0x00 to 0xff, enabling a maximum of 256 TC/LINK-MQs on the same queue set.

5.5.5 Addressing Without Shadow User Search

If you disable the “Search for shadowuser” switch on TC/LINK-MQ (Registry “…\options\UseShadowUser” = “no”), then the following major changes occur in the TC/LINK-MQ operation:

- Message throughput is increased (TC/LINK-MQ does not need to search the user store/address book).
- No originator-dependent templates and coversheets will be inserted.
- All addressing parameters from the transaction file to KCS are inserted “as they are”; empty fields are never filled from a shadow user.
- Addressing parameters are never overwritten (in “Normal” addressing, Full name and some more parameters will be taken from the shadow user, overriding the transaction file values!).
- Addressing to KCS address book is disabled.

In general: If you do not maintain shadow users on KCS, you should use this mode for optimized performance, and to avoid interference with any KCS users.

5.5.6 SSL Between MQ Client and MQ Server

IBM WebSphere can be configured to use SSL for communicating between client and server. Consult your IBM WebSphere documentation for more details. No configuration steps are necessary on TC\LINK-MQ.

An example configuration can be found in the Appendix, section TC/LINK-MQ with SSL.

5.6 Final Installation Steps

Set TCSRV to “startup automatic” (Control Panel / Services / TCSRV).

Restart the TC/LINK-MQ server. This is required to enable SNMP, to replace outdated system DLLs, and to activate the modified system PATH.

5.7 Steps for Testing Configuration

5.7.1 Testing the Local IBM WebSphere MQ Server Configuration

If you installed the local IBM WebSphere MQ server and TC/LINK-MQ, and you have started it for the first time, you can easily check if everything is ok up to now:

Make sure TC/LINK-MQ is running.

Type “AMQSPUT.EXE 2TC.QUEUE TC.QUEUE.MANAGER” from the command line.

This small utility (automatically installed with the IBM WebSphere MQ server; same tool is named “AMQSPUTC.EXE” on the IBM WebSphere MQ client) will first attempt to connect to the queue manager.

If you get a response “target queue is 2TC.QUEUE”, then the connection to the IBM WebSphere MQ server was successful. You can now send a test message to the configured TC/LINK-MQ operator by typing “operator” <Return>. (needs to be lowercase!)

If everything is ok up to now, you will find a greeting message in the operator’s inbox after some seconds.

In the other direction, simply address a message to “MQ,somebody”, and check if it arrives in the “TC2.QUEUE” after a few seconds! Then use the <amqsget> or <MQ2File> utility to extract the message from the queue.

Note: When using the IBM WebSphere MQ client, you can do the same test via the remote IBM WebSphere MQ server; see below.

5.7.2 Testing the Connection from/to the Connected WebSphere MQ Environment

As soon as you were able to send that small test message from a local machine, you can try the same from a remote machine.

Note:

- If you are not using SNA, you need to start a listener on the IBM WebSphere MQ server for Windows (<runmqlsr> command)
- Make sure to start all involved channels and queue managers!

If you get error messages instead, look to section “5.9 Troubleshooting” to find the problem, and for some hints to solve it!

5.8 KCS Test Tools (MQ2File, File2MQ)

Together with TC/LINK-MQ, there are two small command line tools that are installed to the c:\tcoss\tclink\mqtools directory:

- MQ2File.exe can be used to extract an IBM WebSphere MQ message from a queue to a file.
- File2MQ.exe can be used to put a transaction file (plus optional attachment(s)) to an IBM WebSphere MQ queue.

Usage of MQ2File (Message from KCS):

```
C:\TCOSS\TCLINK\mqtools>MQ2File
Syntax is
  "MQ2FILE <Queue name> [<Queue Manager Name> [Codepage]]"

Functionality:
* This tool creates a subdirectory .\TC2 to put all messages from Queue.
* If a transaction file is found on the queue, it is parsed for :ATT: sections.
```

```
* If the file holds :ATT: NAME= sections, then the queue is searched
  for messages with matching CorrelId.
* If found, the attachment is written to the \TC2 directory.
* Finally, the transaction file is also written to the .\TC2 directory.

NOTE:
* This tool is unsupported freeware.
* Messages extracted by this tool will never arrive at the destination!
  THEREFORE, NEVER USE THIS TOOL ON A PRODUCTIVE SYSTEM!
```

Start a command prompt, and change to the <c:\tcoss\tclink\mqtools> directory.

At the first startup of the tool, it creates a <.\TC2> subdirectory.

The tool scans the given queue at the given queue manager. If it finds a message, it is written to the <.\TC2> subdirectory, including the attachments.

```
C:\TCOSS\TCLINK\mqtools>MQ2File TC2.QUEUE TC1.QMGR
Using Client Connection (MQIC32.DLL) - polling ...
MQ2File - Tool to get TCLINKMQ messages from MQ Series Interface

  Polling queue <TC2.QUEUE> at qmgr <TC1.QMGR>, Codepage is 1252

Found transaction message
  Message .\TC2\1031929815.MSG written successfully
Found attachment message
  Message .\TC2\1031929815.01 written successfully
MQ2File Program End
```

Usage of File2MQ.exe (Messages to KCS):

```
C:\TCOSS\TCLINK\mqtools>file2mq
Command-line syntax is:
"FILE2MQ <Queue name> [<Queue Manager Name> [Codepage]]"

Functionality:
* This tool first creates a subdirectory .\2TC
* In this subdirectory, it scans for files with .MSG extension.
* If a file is found, it is sent to the queue/qmgr named at command line.
* If the file holds :ATT: NAME= sections, then the directory is searched
  for files of same name, but with the numeric extension indicated in NAME=

Example:
A file named <test.msg> holds a <:ATT: NAME=1, APPLICATION=x.bmp> line.
Therefore, FILE2MQ will look for a file named <test.1> to put to the queue.

NOTE:
* This tool is unsupported freeware.
* NEVER USE THIS TOOL ON A PRODUCTIVE SYSTEM!
```

Start a command prompt, and change to the <c:\tcoss\tclink\mqtools> directory.

At the first startup of the tool, it creates a <.\2TC> subdirectory.

Now, open notepad (or similar text editor), type in a message in TC Open Message Format, and save it as a <*.msg> file in the <c:\tcoss\tclink\mqtools\2TC> directory (e.g. <test.msg>; the extension is the important thing).

If you do now start the File2MQ tool with proper queue / queue manager parameter, it will take the file and put it to the desired queue.

If the syntax is ok, TC/LINK-MQ will pick the message and send it to KCS.

For any attachments, you must first put the attachments to the <.\2TC> directory, giving them a name of <test.1> for the first, <test.2> for the second attachment, and so on.

In the transaction file (in the text editor), enter a ":ATT: NAME=1, APPLICATION=real.name" sequence for any attachment given.

As soon as you store the text file as <test.msg> (the name must be the same as for all attachments, the extension <msg>!), you can start the File2MQ tool and put the message with all attachments to the indicated queue.

```
C:\TCOSS\TCLINK\mqtools>file2mq 2TC.QUEUE TC1.QMGR
File2MQ - Using Client Connection (MQIC32.DLL)
Putting msgs to <2TC.QUEUE> at qmgr <TC1.QMGR>; Codepage 1252

Reading file .\2TC\testmq_ATT_from_file.msg (size=254)
Reading file .\2TC\testmq_ATT_from_file.1 (size=94208)
Warning: truncated to 1024 bytes
File2MQ Program End
```

Note:

- To get the exact syntax options, type <MQ2File -?> or <File2MQ -?>.
- These tools are unsupported freeware with a major lack in error handling (e.g. files are removed even if putting to MQ is unsuccessful).
- Never use these tools on a productive system! Messages may be lost!
- When sending attachments to KCS by the File2MQ tool, you must use the full format for the :ATT: sections (e.g. ":ATT: NA=1" will not work; you must use ":ATT: NAME=1").
- These tools automatically detect presence of IBM WebSphere MQ server or client (just like TC/LINK-MQ does).

5.9 Troubleshooting

In general, all TC/LINK trace files are located at <c:\tcoss\trace>.

- The TC/LINK-MQ trace files are named "TCLINKMQx.trc".
- The TCSRV trace file is named "BOOTx.trc".

For the IBM WebSphere MQ error logs, you need to check three different locations (assuming installation into c:\mqm):

- <c:\mqm\errors> holds all IBM WebSphere MQ client traces, plus any internal IBM WebSphere MQ errors that occurred (IBM WebSphere MQ "First Failure Symptom" Reports)
- <c:\mqm\qmgrs\@system\errors> holds IBM WebSphere MQ server errors not related to a special queue manager (e.g. incoming connect requests to non-existing queue manager)
- <c:\mqm\qmgrs\TC!QMGR\errors> holds all errors related to the Queue Manager (Note: dots in the queue manager name are replaced by exclamation marks).

All of them are cyclic logs; error logging is - per default - enabled.

5.9.1 TC/LINK-MQ Does Not Start

If you do not get any trace file "TCLINKMQx.trc", but a popup like this.

… then your IBM WebSphere MQ server or client is not installed properly. Uninstall the IBM WebSphere MQ server / client, and continue with section 5.4 "Installing the WebSphere MQ Server on the TC/LINK-MQ "!

- The Windows event-log can give you a quick indication what's wrong.
- If there is already a trace file named "TCLINKMQ0.trc",open it with notepad.exe, and look inside for any failure codes and error details.
- For getting an output of all IBM WebSphere MQ queue properties, set the …\General\Tracelevel to 0x100 (256 dec)!

Attention: All IBM WebSphere MQ object names (queue manager, queue names, connection channels, …) are basically case-sensitive! So, it is a good practice to write them in UPPERCASE letters always.

5.9.2 Dump IBM WebSphere MQ Configuration

Setup copies a file "DUMP.TXT" to the MQTOOLS directory. If you encounter problems, you can use this file to dump the complete IBM WebSphere MQ configuration for the Queue Manager used by TC/LINK-MQ.

Copy the file to the IBM WebSphere MQ server and type

```
"runmqsc TC.QUEUE.MANAGER <DUMP.TXT  >DATA.TXT".
```

The configuration dump is then written to file DATA.TXT.

Contents of DUMP.TXT:

```
DISPLAY QMGR all

DISPLAY QLOCAL (*) all
DISPLAY QALIAS (*) all
DISPLAY QMODEL (*) all
DISPLAY QREMOTE (*) all
DISPLAY QUEUE (*) all

DISPLAY CHANNEL (*) all
DISPLAY CHSTATUS (*) all

DISPLAY PROCESS (*) all
```

5.9.3 Most Common IBM WebSphere MQ Error Codes, And Possible Reasons

This section gives a short overview about the most important IBM WebSphere MQ error codes. To find a complete list of possible errors, refer to IBM documentation!

MQ error code	Description	Possible reasons/workarounds
2034	No message under cursor	This is a non-fatal message that may occur if multiple TC/LINK-MQs work on the same set of queues. No action required.
2035	Not authorized	Windows User ID must not be longer than 12 characters! When using local IBM WebSphere MQ server: Windows User ID must be member of either mqm or Administrator group! When using IBM WebSphere MQ client: Make sure that a security profile exists for the Windows User ID on the IBM WebSphere MQ server system! With separate security system (e.g. RACF on MVS): make sure that appropriate security profiles are defined for any accessed objects (queues, storage, …)!
2042	Object in use	Make sure that all applications use TC/LINK-MQ queues in shared mode If you are using remote queues, make sure to enter the remote queue definition name to the TC/LINK-MQ queue name setup – not the transmission queue name!
2046	Invalid options	Check if you are trying to use a remote queue as 2TC.QUEUE / DIRSYNC.QUEUE! These two must be QLOCAL! Check if your application uses any unsupported options (e.g. MQGMO_BROWSE_MSG_UNDER_CURSOR on MVS)
2053	Queue full	Check if connected application is running, and has sufficient throughput Check queue configuration for proper MAXDEPTH setting
2058	Queue Manager Name error	Check the TC/LINK-MQ configuration for correct Queue manager name When using AMQCLCHL.TAB: verify that the correct QMNAME

		has been defined on the server; verify that you have the latest AMQCLCHL.TAB file from the server (must be copied to the client any time the server channel configuration changes!)
2059	Queue Manager not available	Check if local queue manager is running (try runqmsc <qmgr>) Check if involved channels are started Check if transport-dependent environment works (e.g. listener is started on Windows)
2085	Unknown object name	Check the TC/LINK configuration for correct Queue names Check involved channel – and remote queue definitions for correct transmission queue names
2102	Resource problem	Check local hard disk space With IBM WebSphere MQ client: check IBM WebSphere MQ server disk space Check virtual memory (swap file size)
2161	Queue manager quiescing	Queue manager is currently shutting down. Retry later.
2288	Unknown queue name	Make sure to enter the correct queue names in TC/LINK-MQ setup.

5.9.4 Testing the Network Transport Protocol

5.9.4.1 General Testing

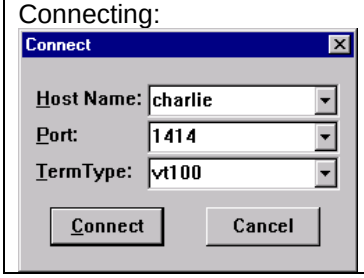
On the IBM WebSphere MQ server, start <runmqsc>, and type <ping channel(<your channel name>)>. If the output is successful, the channel is running and ok. If you get something like “remote channel is unavailable”, you need to make sure that the channel is started, and appropriate listeners are running.

5.9.4.2 TCP Transport

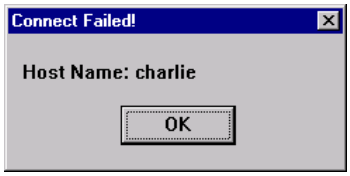
MQ uses TCPIP port 1414 by default. You can try the following:

- Type <ping <remote host name>> from the command line. If successful, name resolution and IP connectivity is ok.
- With telnet, connect to port 1414 (IBM WebSphere MQ default port number) of the remote host name:

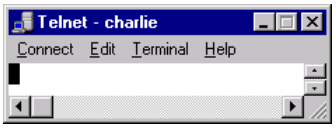
Connecting:



Failure:



Success:



If (after some seconds) the connection fails, then the listener is not running on the destination host.

If you connect successfully (note the black telnet cursor), then the connection was successful.

Note: The IBM WebSphere MQ listener must be bound to a specific queue manager (e.g. on Windows, by the <runmqlsr –m <Queue Manager>> option)! Therefore, if the connect is successful, but you cannot get any data transferred, check the listener options!

5.9.5 Additional Hints

In the system default queue “SYSTEM.ADMIN.CHANNEL.EVENT”, you will find an entry for any channel events (anytime channel starts/stops, any errors). This may be useful for problem determination.

To view this queue, there are several tools available; e.g. the “PQEdit” from Candle Corp. (see 8.2 “Further Sources of Information”).

6. Performance

TC/LINK-MQ has a throughput of minimum 2000 messages (single-page text) per hour on the following environment:

- TC/LINK-MQ and IBM WebSphere MQ server 5.0 (CSD 04) running on Model 21x Link server, Pentium 200, 64MB RAM, Windows NT 4.0 (SP 4)
- TCOSS is running on Model 21x TCOSS server; no other TCOSS load.
- TCP connections, 100Mbit Ethernet LAN

This performance data can – of course – be degraded by slow client-server connections, overloaded IBM WebSphere MQ servers, poor server hardware, or similar. To get optimum performance, the following is recommended:

- Use TCP/IP transport
- Avoid frequent connects/disconnects in the connected application (TC/LINK-MQ does – except for dead-letter queue polling - a single connect at startup, and a single disconnect on shutdown).
- Avoid any intermediate storage in local queues
- Use high batch size in channels (BATCHSZ parameter in channel definition shall be at least 50)

7. Restrictions

See the TC/LINK-Manual for all general TC/LINK restrictions!

IBM WebSphere MQ message size limits apply (configuration and hardware-dependent; e.g. max. 4 Mbytes on an IBM WebSphere MQ 4.x server, 100 Mbytes on IBM WebSphere MQ 5.0).

No support for embedded messages (embedded messages from KCS are converted to flat messages, keeping all content)

To route messages or notifications to a specific host application (or instance of an application), an instance of TC/LINK-MQ (plus separate queue set) is needed per application instance.

Use of any IBM WebSphere MQ exits on KCS servers is allowed, but not supported. If any problems occur, the exits must be removed.

Parallel installation (with identical configuration) on same Windows computer is not supported.

Japanese installations (codepage 932) are only supported to send/receive fax messages; Rome compliance is not supported with Japanese codepage.

IBM WebSphere MQ client 2.1 for Windows is definitely not supported; minimum IBM WebSphere MQ client version is 5.0.

MQ Message Segmentation does not work with character conversion. Character conversion is only performed for TOM transaction files if requested by the application calling MQGet.

7.1 Restrictions in TC/XML Functionality

Only TCOSS 0 and 1 code pages are supported. Other code pages like 932 will be supported on request.

8. Appendix

8.1 Special Features

If you went through till here, you know all the important issues for sending and receiving messages. The following sections will mention some special features that may be useful in some environment, but are not generally required for operation.

8.1.1 Notifications from KCS

By default, notifications from KCS are put to a separate queue ("NOTIF.QUEUE") to allow for different handling (lower priority handling, or special notification handler applications).

If you want your notifications on the same queue as the messages, you can also configure the notification queue to be the same as the "TC2.QUEUE".

In order to match the notifications to the original transaction file, applications may use the C1…C5 fields of every recipient (TO: section in the original transaction file). These textual fields are returned unchanged in the corresponding notification (NFINFO: section). See the "TC Open Message Format manual" for details and examples.

8.1.2 Directory Synchronization

TC/LINK-MQ supports directory synchronization in the direction to KCS. Any application can generate so-called "dirsync-messages" to add, modify or delete users or address book entries on KCS.

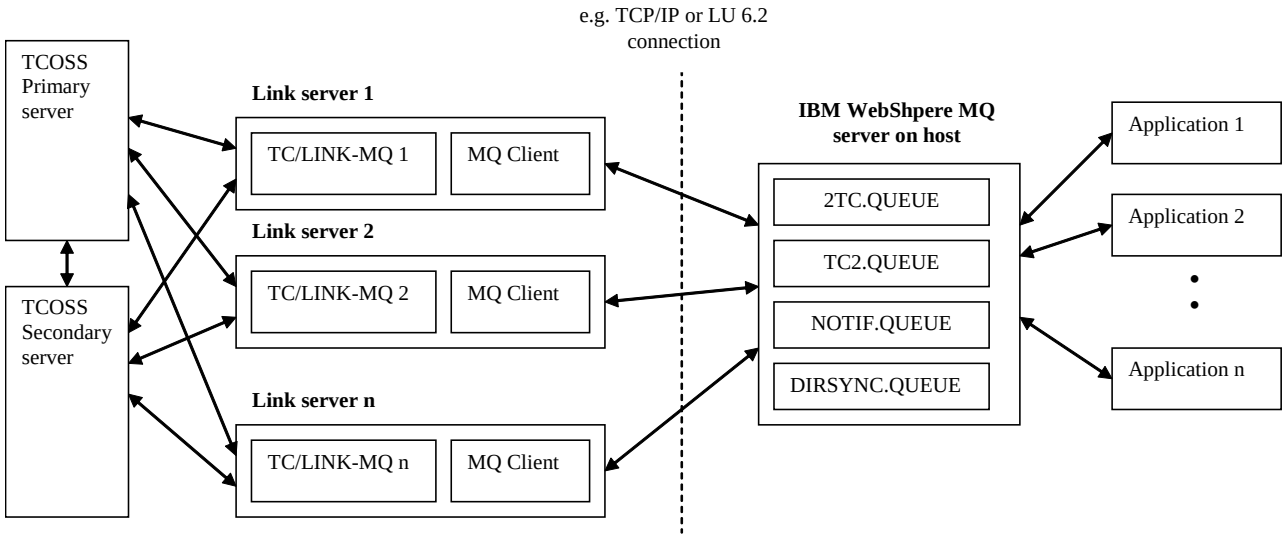
For reasons of security, these messages are by default handled by a separate queue ("DIRSYNC.QUEUE"). If you don't need that feature, you can also configure the dirsync queue to be the same as the "2TC.QUEUE".

For a detailed description of the dirsync message format, see the "TC Open Message Format" manual.

For a detailed description of the dirsync operation, see the general TCLINK manual.

8.1.3 Automatic Fail-Over and Load Balancing

Concept:



In order to build a fail-safe system (at least on the KCS side), you need to do the following:

- You need a tandem TCOSS server
- At least two TC/LINK-MQs must be installed on separate Windows computers.
- All of these Links connect to the same tandem TCOSS server (make sure to configure the alternative paths in the TC/LINK-MQ advanced setup)

- All of these Links need a local IBM WebSphere MQ client.
- The use of a local IBM WebSphere MQ server is NOT allowed for automatic fail-over (the local server buffers all messages on local queues; in case of system failure, these messages would hang in the local queues!)
- You must connect the local IBM WebSphere MQ client directly to the final destination queue (e.g. on host); any intermediate queues again give the risk of hanging messages!
- You must use syncpoint control with the connected application.

See the general TCLINK manual for details on configuration of multiple links in parallel on separate workstations!

Fail-over scenario:

- As long as everything works fine, load is shared between all Link servers and applications.
- If a Link server goes down while idle, message transmission will remain unchanged through the other Links. So, beside reduced maximum throughput, there is a fully transparent fail-over.
- If a Link goes down while receiving a message from MQ, the message will automatically be backed out due to the non-graceful disconnect from MQ. The next time any of the other links polls the MQ server, it will take that message.
- If a Link server goes down while transmitting a message to MQ, this message will stay on “sending” status for one hour; after one hour, it is sent via one of the remaining links. So, there is a maximum delay of one hour.
- If appropriate, KCS will set the “possible duplication” flag of the message.
- If one of the TCOSS servers goes down, all Links will transparently switch to the other server of the tandem TCOSS system.

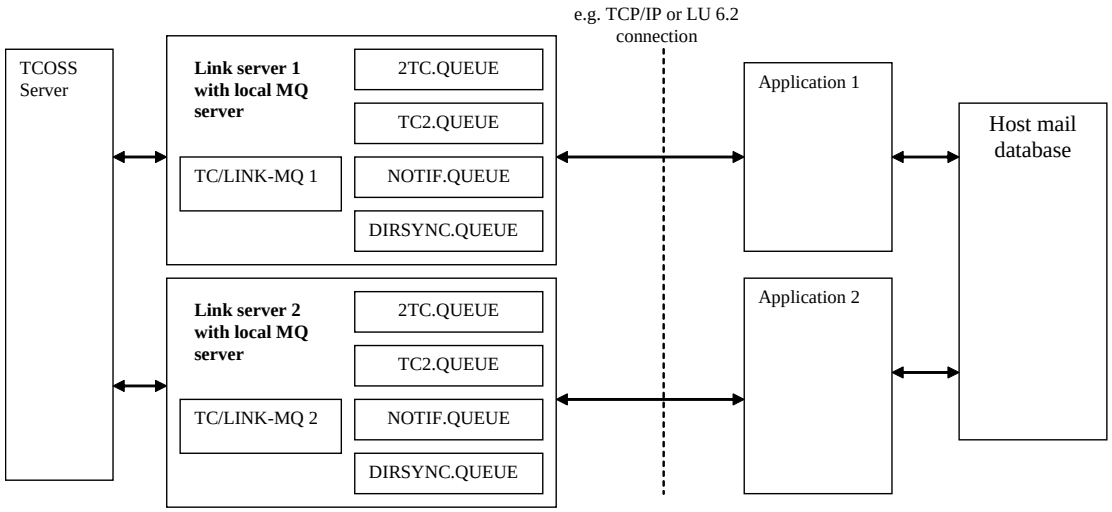
Note: If the IBM WebSphere MQ server on the host fails, the whole system is down (that is why I wrote “at least on the KCS side” …): The connected applications will no longer be able to work; messages that are already in the host queues are blocked until the system is up again.

8.1.4 Load Balancing for Increased Throughput (No Failover)

If automatic fail-over is not a concern, but you need maximized performance, you can use a similar scenario.

- You can connect multiple links to the same KCS server
- Those TC/LINK-MQs will share all the message traffic from KCS.
- You are allowed to use intermediate local queues (e.g. on local IBM WebSphere MQ server). Using a local IBM WebSphere MQ server will result in optimum performance.
- If you use local queues, your application must be able to work in parallel (any application must also be capable of handling notifications for all other).

Example scenario with local queues:



Note:

- To increase throughput, all connected TC/LINK-MQs must be installed on separate Link servers!
- Each of the Link servers requires the local IBM WebSphere MQ Server or client.
- On failure of any component, there is the risk for messages hanging in any queue! So, this scenario is NOT a fail-save design!

8.1.5 Connecting via MQ-Client to Multiple Queue Managers

Multiple instances of TC/Link-MQ run on one Link-Computer. The MQ-Client is installed to connect to the queue managers on one or multiple MQ-Servers.

The IBM MQ-Client manual states in Chapter 12: “Running applications on IBM WebSphere MQ clients”:

“Define your client-connection and server-connection channels on one queue manager only, including those channels that connect to a second or third queue manager. Do not define them on two queue managers and then try to merge the two client channel definition tables; this cannot be done. Only one client channel definition table can be accessed by the client.”

That means that there has to be one queue manager that has all necessary client connection channels defined, also those to the other queue managers.

8.1.5.1 Example with Two TC/Link-MQ Instances

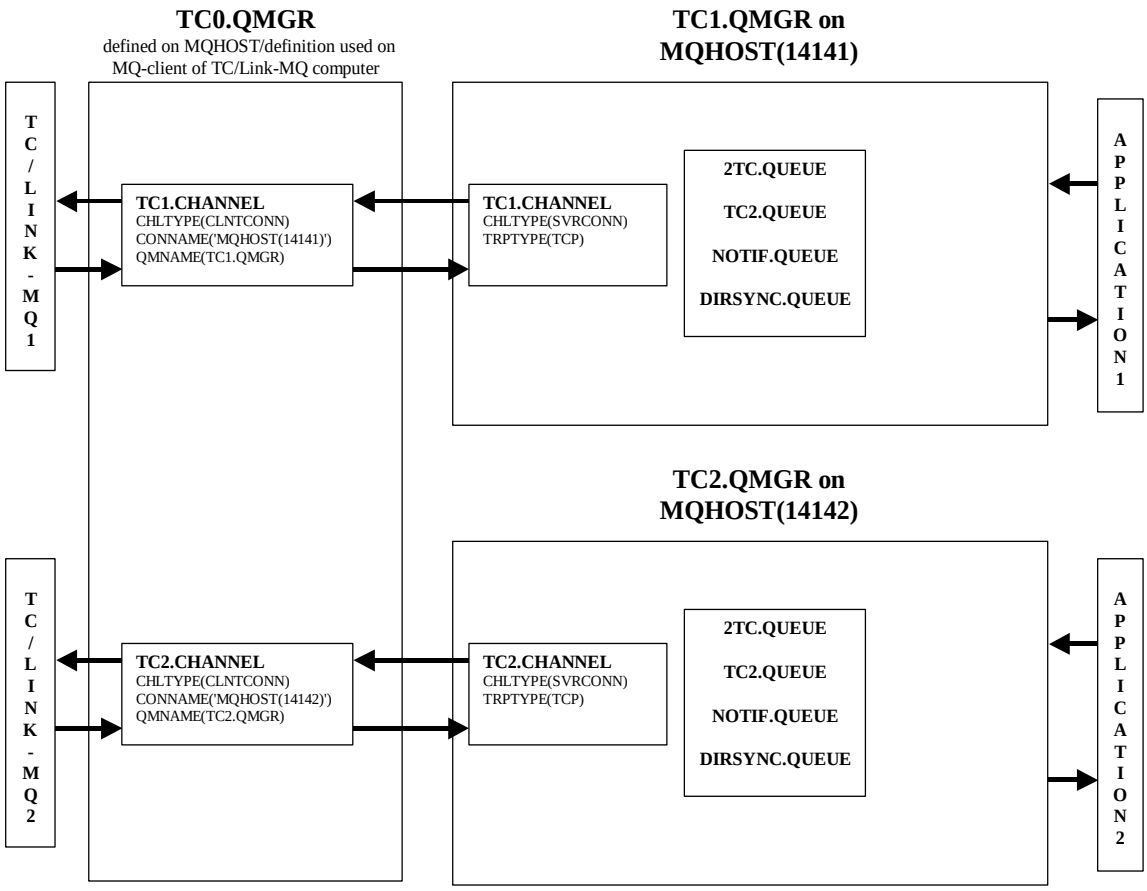
The following example shows how the MQ-environment can be configured for two TC/Link-MQ instances.

A dedicated queue manager (TC0.QMGR) is used for the definition of the two client connection channels. These channels point to the queue managers administering the TC queues. The used transport is TCP/IP; both queue managers are running on a computer called MQHOST, but both have listeners using different port-numbers.

The TC Queue Managers (TC1.QMGR and TC2.QMGR) have the corresponding server connection channels defined and a set of the standard TC queues. On the other side the 3rd party applications have to be connected to the queues with appropriate MQ methods.

After the definitions of the TC0.QMGR have been made, the “Amqclchl.tab” file (holding the definitions of the queue manager, on an Windows-computer found typically at "c:\Program Files\MQSeries\qmgrs\TC0!QMGR\@ipcc\") is put on the TC/Link-MQ computer to the main MQ folder (“c:\MQM” or “c:\Program Files\MQSeries”). TC/Link-MQ asks at startup the MQ-Client if it is possible to connect to the TC Queue Manager. The MQ-Client finds the information for connection in the “Amqclchl.tab” file and is able to connect. If the file is not found or the information is not correct, startup of TC/Link-MQ fails. The queue manager TC0.QMGR however is no longer necessary for the connection. We needed it only to generate the “Amqclchl.tab” configuration file.

The following figure illustrates the described configuration.



8.1.5.2 Queue Manager Definitions

The queue managers TC0.QMGR, TC1.QMGR, TC2.QMGR have to be created and started. For each of TC1.QMGR and TC2.QMGR also a listener has to be created and started. In the example both TC1 and TC2.QMGR are on the same computer, as transport TCP/IP is used, so the listeners are configured to use different port-numbers (14141 and 14142).

To define the necessary channels and queues the following text files can be used. There is one for each of the queue managers. MQSetup_TC1.txt and MQSetup_TC2.txt are identical except for the name of the server connection channel.

MQSetup_TC0.txt

```
********************************************************
*   Command syntax:
*     runmqsc qmgrname < Mqsetup_TC0.txt
*
*
*   This is the definition file of the Queue Manager
*   that defines the client connections.
*
********************************************************

********************************************************
*
* For each Link-MQ instance/Queue Manger one pair of
* Server/Client channels have to exist; the client
* connections are all defined on one Queue Manager,
* which definitions are later transferred via the
* Amqclchl.tab definition file to the MQ-Client;
* the matching server connections are defined
* on the Queue Managers that administer the TC queues
*
```

```
*******************************************************

    DEFINE CHANNEL (TC1.CHANNEL) +
           CHLTYPE(CLNTCONN) +
           TRPTYPE(TCP) +
           CONNAME('MQHOST(14141)') +
           QMNAME(TC1.QMGR) +
           DESCR('Client Connection to TC Queues')
    DEFINE CHANNEL (TC2.CHANNEL) +
           CHLTYPE(CLNTCONN) +
           TRPTYPE(TCP) +
           CONNAME('MQHOST(14142)') +
           QMNAME(TC2.QMGR) +
           DESCR('Client Connection to TC Queues')
```

MQSetup_TC1.txt

```
*******************************************************/
*   Command syntax:                                   */
*     runmqsc qmgrname < Mqsetup_TC1.txt              */
*                                                     */
*                                                     */
*   This is the definition file for the KCS           */
*   server environment.                               */
*   All maximum message sizes are set to 4MB in this  */
*   sample file. Edit it according to your needs.     */
*                                                     */
*******************************************************/

*******************************************************/
* LOCAL QUEUE DEFINITIONS                             */
*******************************************************/

    DEFINE QLOCAL('2TC.QUEUE') +
     DESCR('Message Queue from MQ to KCS') +
     PUT(ENABLED) +
     DEFPRTY(0) +
     DEFPSIST(YES) +
     MAXMSGL(4000000) +
     GET(ENABLED)

    DEFINE QLOCAL('TC2.QUEUE') +
     DESCR('Message Queue from KCS to MQ') +
     PUT(ENABLED) +
     DEFPRTY(0) +
     DEFPSIST(YES) +
     MAXMSGL(4000000) +
     GET(ENABLED)

    DEFINE QLOCAL('NOTIF.QUEUE') +
     DESCR('Notification Queue from KCS to MQ') +
     PUT(ENABLED) +
     DEFPRTY(0) +
     DEFPSIST(YES) +
     MAXMSGL(4000000) +
     GET(ENABLED)

    DEFINE QLOCAL('DIRSYNC.QUEUE') +
     DESCR('Directory Synchronization Queue from MQ to KCS') +
     PUT(ENABLED) +
     DEFPRTY(0) +
     DEFPSIST(YES) +
     MAXMSGL(4000000) +
     GET(ENABLED)

*******************************************************/
* A SINGLE SERVER/CLIENT CHANNEL DEFINITION PAIR TO    */
```

```
* ALLOW CLIENTS FROM OUTSIDE TO CONNECT TO MY QUEUES   */
*
* The client connections are all defined on one Queue
* Manager, which definitions are later transferred via
* the Amqclchl.tab definition file to the MQ-Client;
* the matching server connections are defined here,
* on the Queue Manager that administers the TC
* queues
*
*********************************************************/
    DEFINE CHANNEL (TC1.CHANNEL) +
           CHLTYPE(SVRCONN) +
           TRPTYPE(TCP) +
           DESCR('Server channel for client connections to TC queues')

*********************************************************/
* Some definitions that are usually already there.     */
* However, by default, recreate them if deleted!       */
*********************************************************/

    DEFINE QLOCAL('SYSTEM.DEAD.LETTER.QUEUE') +
     DESCR('IBM WebSphere MQ default dead queue')

    ALTER QMGR DEADQ(SYSTEM.DEAD.LETTER.QUEUE)
```

MQSetup_TC2.txt

```
*********************************************************/
*   Command syntax:                                     */
*     runmqsc qmgrname < Mqsetup_TC2.txt                */
*                                                       */
*                                                       */
*   This is the definition file for the TC              */
*   server environment.                                 */
*   All maximum message sizes are set to 4MB in this    */
*   sample file. Edit it according to your needs.       */
*                                                       */
*********************************************************/

*********************************************************/
* LOCAL QUEUE DEFINITIONS                               */
*********************************************************/

    DEFINE QLOCAL('2TC.QUEUE') +
     DESCR('Message Queue from MQ to KCS') +
     PUT(ENABLED) +
     DEFPRTY(0) +
     DEFPSIST(YES) +
     MAXMSGL(4000000) +
     GET(ENABLED)

    DEFINE QLOCAL('TC2.QUEUE') +
     DESCR('Message Queue from KCS to MQ') +
     PUT(ENABLED) +
     DEFPRTY(0) +
     DEFPSIST(YES) +
     MAXMSGL(4000000) +
     GET(ENABLED)

    DEFINE QLOCAL('NOTIF.QUEUE') +
     DESCR('Notification Queue from KCS to MQ') +
     PUT(ENABLED) +
     DEFPRTY(0) +
     DEFPSIST(YES) +
     MAXMSGL(4000000) +
     GET(ENABLED)
```

```
    DEFINE QLOCAL('DIRSYNC.QUEUE') +
     DESCR('Directory Synchronization Queue from MQ to KCS') +
     PUT(ENABLED) +
     DEFPRTY(0) +
     DEFPSIST(YES) +
     MAXMSGL(4000000) +
     GET(ENABLED)

*********************************************************/
* A SINGLE SERVER/CLIENT CHANNEL DEFINITION PAIR TO    */
* ALLOW CLIENTS FROM OUTSIDE TO CONNECT TO MY QUEUES   */
*
* The client connections are all defined on one Queue
* Manager, which definitions are later transferred via
* the Amqclchl.tab definition file to the MQ-Client;
* the matching server connections are defined here,
* on the Queue Manager that administers the TC
* queues
*
*********************************************************/
    DEFINE CHANNEL (TC2.CHANNEL) +
           CHLTYPE(SVRCONN) +
           TRPTYPE(TCP) +
           DESCR('Server channel for client connections to TC Queues')

*********************************************************/
* Some definitions that are usually already there.     */
* However, by default, recreate them if deleted!       */
*********************************************************/

    DEFINE QLOCAL('SYSTEM.DEAD.LETTER.QUEUE') +
     DESCR('IBM WebSphere MQ default dead queue')

    ALTER QMGR DEADQ(SYSTEM.DEAD.LETTER.QUEUE)
```

8.1.5.3 TC/Link-MQ Configuration

On KCS side two instances of TC/Link-MQ are installed. The configuration of both is identical except for the following:

On KCS two different services and queues have been created for the TC/Link-MQ instances. This leads to the following registry entries:

```
[HKLM\SOFTWARE\Topcall\TCLINKMQ1\Setup\ServiceMQ]      […\TCLINKMQ2\Setup\ServiceMQ]
"Name"="MQ1"                                          "Name"="MQ2"
"Prefix"="TCLMQ1I:"                                   "Prefix"="TCLMQ2I:"
[HKLM\SOFTWARE\Topcall\TCLINKMQ1\Topcall]             […\TCLINKMQ2\Topcall]
"Queue"="TCLMQ1"                                      "Queue"="TCLMQ2"
```

For connecting to the appropriate queue managers, the following has to be configured (the queue names remain unchanged with the default values):

```
[HKLM\SOFTWARE\Topcall\TCLINKMQ1\Options]       […\TCLINKMQ2\Options]
"QueueManager"="TC1.QMGR"                       "QueueManager"="TC2.QMGR"
```

8.1.6 Maximum Queue Depth

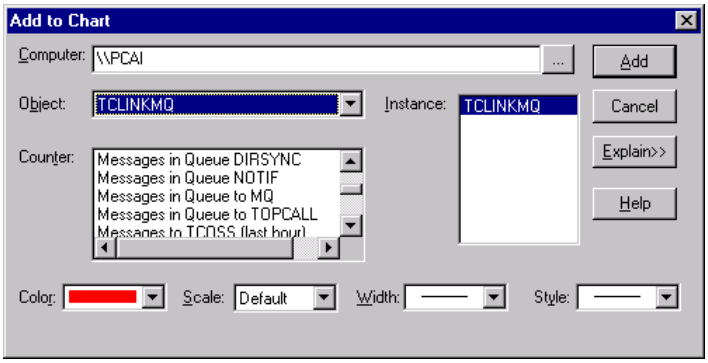
TC/LINK-MQ checks the queue's maximum depth at startup (MAXDEPTH queue property). TC/LINK-MQ does not put further messages to the queue if this value is reached. Retries are performed after some time.

Polling the "2TC.QUEUE" and "DIRSYNC.QUEUE" remains unchanged, even with "queue full" condition on any of the connected queues.

8.1.7 Special TC/LINK-MQ Performance Counters

TC/LINK-MQ uses four special performance counters to display the current depth of any of the four available queues.

Performance monitor selection screen:



Note:

- Queue depth can only be inquired from local queues (QLOCAL)! When using remote queues (QREMOTE), the corresponding counters always display 0.
- Performance counters work with IBM WebSphere MQ server or client. In case of client, the current depth of the queues on the connected server is displayed.
- The displayed queue depth indicates the number of IBM WebSphere MQ messages, which can be different from the number KCS messages (as described 3.1.2 “Basic Message Design”, a KCS message may consist of multiple IBM WebSphere MQ messages). Watch out: TCMON always displays the number of KCS message sent/received.

See the TCLINK manual for a description of all generally available performance counters.

8.1.8 TCDC, SNMP and More

As TC/LINK-MQ is built on the TC/LINK architecture, all common TC/LINK features are available for all MQ-enabled platforms. See the general TCLINK manual for a complete overview of features!

8.1.9 Code Page Conversion for All Attachments

This version checks at startup for the DWORD registry key …\Options\AttachmentConvert. If this key exists and is set to 1 TC/LINK-MQ will convert not only the transaction files but also all attachment files it reads from the queue according to ...\General\PCCodePage. It does NOT check for the extensions of attachment files (as described in enhancement 7326). If there are any binary files sent with AttachmentConvert set, the files will most likely be either corrupted or rejected.

Registry Keys (HKLM\Software\Topcall\TCLINKMQ\Options):

Registry Key	Type	Default	Description
AttachmentConvert	DWORD	0	0 - No conversion of Attachments 1 - Convert Attachments to configured code page

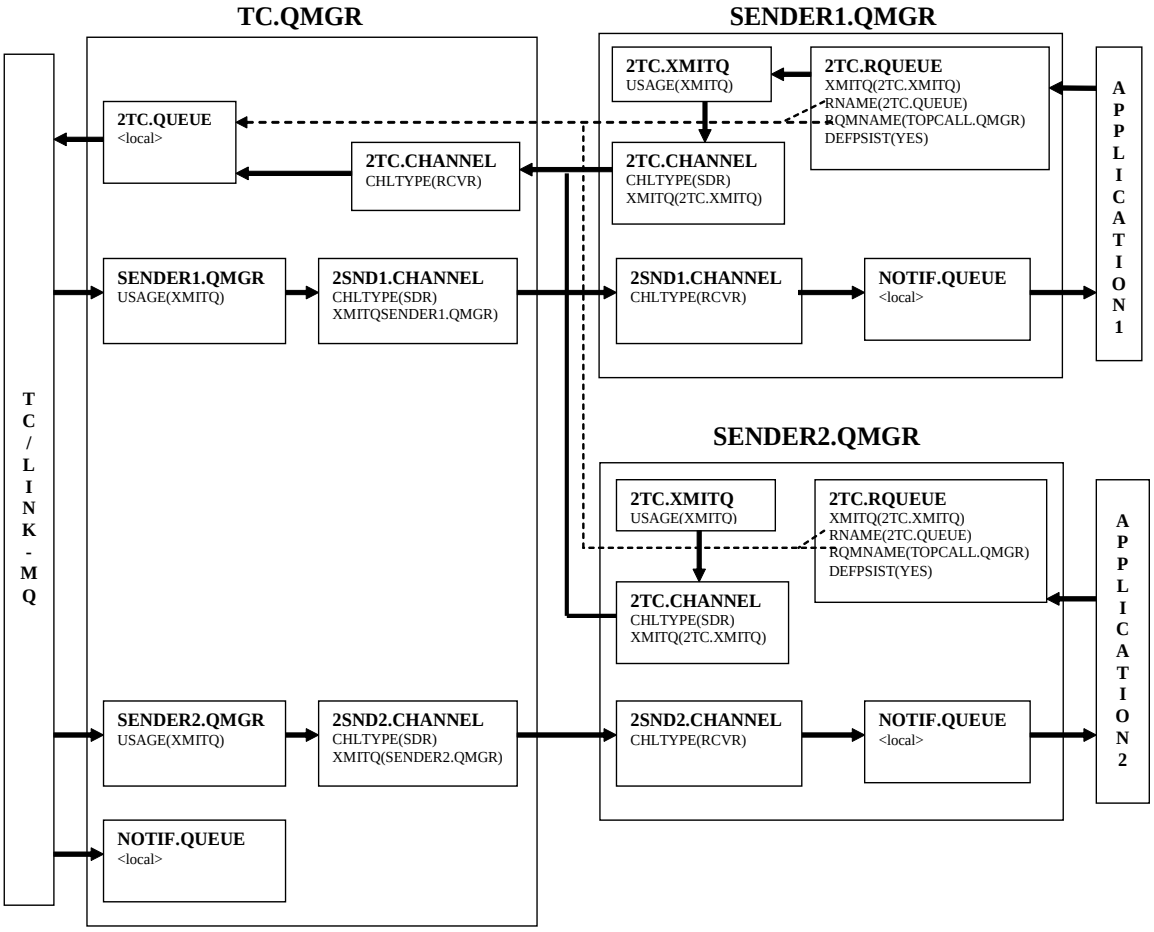
8.1.10 Notifications to Reply-To Queue Manager

The MQ Message Descriptor (MQMD) sent with each message provides information of the original Queue Name and Queue Manager Name a message came from. If a notification is returned for a message, it is important to send it back to the correct Queue and Queue Manager, so that large MQ environments can route the message correctly to the originator.

To support that routing mechanism TC/LINK-MQ reads the Reply-To Queue Manager Name from an incoming message and stores it to one of the recipient specific correlation parameters (C1 - C5). When a notification is sent back, TC/LINK-MQ will check that correlation parameter, look if a transmission-queue with the name of the Reply-To Queue Manger Name exists and put it there.

The MQ system has to be configured that for each Reply-To Queue Manager there has to exist a path back to a queue where the notification is handled, starting with a transmission-queue on the TC Queue Manager. This transmission-queue has to have the name of the Reply-To Queue Manager Name of the incoming message. If no queue of this name exists, the notification is put to the standard NOTIF Queue. On each of the Reply-To Queue Managers there has to exist a NOTIF Queue with the same name as on the TC Queue Manager.

The following figure shows an example for an MQ configuration. Application1 and Application2 are applications sending messages to KCS. TC/LINK-MQ reads the messages with the ReplyToQmgr from the MQMD put to the 2TC.QUEUE. Notifications for messages from the Queue Manager SENDER1.QMGR are put to the transmission queue SENDER1.QMGR, that from SENDER2.QMGR are put to the queue SENDER2.QMGR. The necessary transmission queues and channels have to be configured for each Queue Manager sending to KCS that require a notification path back to a special queue.



The following registry key makes it possible to configure the correlation field that is used on KCS side to store the MQMD information.

Registry Keys (HKLM\Software\Topcall\TCLINKMQ\Options):

Registry Key	Type	Default	Description
StoreReplyToInCorr	DWORD	0	Number of correlation parameter that is used for storing Reply-To information (Queue Manager Name) of the MQMD for returning of notifications from KCS. 0 Disabled, Reply-To is ignored and notifications go to local NOTIF Queue. Number of correlation parameter (for C1-C5 of recipient). A notification is put to a queue of the name of the Reply-To Queue Manager Name.

The configured correlation field must not be used by the sending application. If the correlation field is used or the key is set to a wrong value, no Reply-To data will be stored and the notification will be put to the local NOTIF-Queue. If the key is set to 1-5 and there is no Queue defined for the Reply-To Queue Manager Name of an arriving message the notification will be put to the local NOTIF-Queue.

Here is an example of a message sent from an application to KCS. Registry key StoreReplyToInCorr is set to 5, which means that the C5 correlation parameter is set to the value of ReplyToQmgr.

MQGET from standard in-queue: “2TC.QUEUE”

MQMD: ReplyToQmgr = “SENDER1.QMGR”

```
SUBJECT=”Notification to Original Queue Manager”, NF=ALL
FROM: SERVICE=MQ, NUMBER=MeOnMQ, SN=MeOnMQ
TO: SERVICE=FAX, NUMBER=66133831, C1=4711, C2=MyFirstMessageWithCorrel
TXT:
Some test message
```

C5 = “SENDER1.QMGR”

The notification that would be returned:

MQOPEN/MQPUT to queue NOTIF.QUEUE on queue manager SENDER1.QMGR as defined in C5

```
SUBJECT="Del: Notification to Original Queue Manager", TYPE=NOTIF, MCORR=00001639982
FROM: ACTIVE=YES, SERVICE=FAX, NUMBER=66133831
TO: ACTIVE=YES, SERVICE=TCMQ, NUMBER=MeOnMQ
NFINFO: STATUS=DEL, TIME=010518:143100, MCORR=00001639982, C1=4711,
C2=MyFirstMessageWithCorrel, C5=SENDER1.QMGR

TXT:
TOPCALL Delivery Notification
-----------------------------

Message 00001639982 successfully sent to FAX,66133831
Time sent: 01-05-18 14:31:00
Subject: TOPCALL Open Message Format Description
Cost: 5 for GUEST
```

Calling MQOPEN and specifying a remote queue manager (SENDER1.QMGR) works only if there is a transmission queue defined with exactly the name of the remote queue manager. For a correct delivery of the message there have also the NOTIF.QUEUE on the remote queue manager and the channels for transferring the message to be defined.

8.1.11 MQ Message Segmentation

TC/LINK-MQ supports message segmentation and reassembly by application. This is for messages (transaction files or attachments) that are too large for the queue manager or the queue to handle.

To activate this feature, you have to set the following registry key to 1:

HKEY_LOCAL_MACHINE\SOFTWARE\TOPCALL\TCLINKMQ\Options

Registry Value	Type	Default	Description
MessageSegmentation	DWORD	0	Enable support for MQ message segmentation. If set to 1, messages that are too large for the queue manager or queue to handle are segmented and reassembled by TC/LINK-MQ.

It might be necessary to increase the log file size and numbers of the queue manager.

MQ Message Segmentation does not work with character conversion. Character conversion is only performed for TOM transaction files if requested by the application calling MQGet.

8.2 Further Sources of Information

8.2.1 On the Local IBM WebSphere MQ Server

With standard IBM WebSphere MQ server installation, you will find a complete set of html documentation in the directory c:\mqm\books. IBM WebSphere MQ also creates a program shortcut in the start menu; follow “Start” -> “Programs” -> “IBM WebSphere MQ for Windows” -> “IBM WebSphere MQ online documentation”.

If you did not install the documentation to the hard disk, you can also browse the IBM WebSphere MQ installation CD (documentation is found in <\books\html> on the IBM WebSphere MQ Server CDROM)

8.2.2 On the Internet

Another good hint is to visit the IBM WebSphere MQ homepage at http://www.software.ibm.com/ts/MQSeries/.

There you find information about latest releases, available PTFs (“Program temporary fix”), CSDs (“Cumulative Service Diskettes”) ,lots of sample code, MQ clients for free download, some utilities (like a remote maintenance client – search for “MO71”), and more.

For troubleshooting, it may be helpful to search the IBM WebSphere MQ mailing list archives at http://www.infochain.be/framesets/start.html

If you have to deal regularly with IBM WebSphere MQ issues, you may also want to subscribe to the mailing list: Simply send an email with a text of “subscribe mqseries@AKH-WIEN.AC.AT” to the address <LISTSERV@AKH-WIEN.AC.AT>

Nice tools for Windows-IBM WebSphere MQ administration are available e.g. from www.candle.com.

8.3 Glossary / Abbreviations

IBM WebSphere MQ in general:

- CCSID … Coded Character Set ID
- CICS … Customer Information Control System (a transaction monitor)
- CSD … Cumulative Service Diskettes (… Microsoft calls the same thing “Service pack”)
- IMS … Information Management System (an operating system on top of MVS)
- MQ … Message Queuing
- MS-MQ … an IBM WebSphere MQ-like API provided by Microsoft. However, it is propriety, only supported on Windows environment, and not compatible to the IBM WebSphere MQ (therefore, also not compatible to TC/LINK-MQ)! However, there are gateways available that connect the two MQ systems.
- MVS/ESA … Multiple Virtual Storage/Enterprise Systems Architecture; the old operating system for host/mainframe computers.
- OS/390 … the actual operating system for host/mainframe computers (P/390 machines).
- PTF … Program Temporary Fix (… Microsoft calls the same thing “Hotfix”)

Terms used in IBM WebSphere MQ Messages/Reports:

- COD … Confirmal of delivery
- COA … Confirmal of Arrival
- PAN … Positive Action Notification
- NAN … Negative Action Notification
- PMO … Put Message Options
- GMO … Get Message Options
- MQMD … IBM WebSphere MQ Message descriptor

SNA-related:

- APPN … Advanced Peer-to-Peer Networking
- CP … Control point
- CPI-C … Common Programming Interface for Communication

- LEN … low-entry networking node
- LU … Logical unit
- NAU … network accessible units
- SAP … Service Access Point
- SNA … Systems Network Architecture
- SSCP … system services control point
- TP … Transaction program

8.4 XML Specific

8.4.1 Code Page Conversion

The TCOSS code page (0 or 1) can be configured during setup or in the registry. The PC code page is not used with the XML format; instead XML specific encoding is used.

Incoming XML messages define for themselves the used encoding. The encoding of an outgoing XML message is defined either by the transformation style sheet, or if no transformation is used by the "…\Options\Out\XMLEncoding" registry parameter.

8.4.2 Default Values

For incoming messages (Mail to KCS) default values are set like for the TOM format. The following TCSI parameters are set if they are missing:

SET_HEADER Object:

Object	Default	Description
INT_TERMINATION	NOTIF_NEG	Flags if Notifications for Non-Delivery and Delivery have to be returned, the default is for Non-Delivery only
SET_ENTRY_RS_ ORIGINATOR	Taken from registry	Originator of the message, default is taken from the registry (…TCLINKMQ\Options\DefaultOriginatorService and …DefaultOriginatorNumber)

L_RECIPIENT Object:

Object	Default	Description
INT_ACTIVE	YES	The recipients have to be set to active
INT_DEL_TYPE	TO_	Delivery Type: To, Cc, Bcc, or Authorize

8.4.3 XML Tools

Two tools are copied by setup to the C:\TCOSS\TCLINK\xmltools directory. None of these tools is officially supported by KCS but may help to troubleshoot.

MSXLS.exe is a standard Microsoft tool that performs XSLT transformation. The input parameters are a XML file and a XSLT style sheet that defines the transformation. This tool can be used for testing a customized XSLT transformation style sheet or to transform between TC/XML and TCXL.

XML2tc.exe is a KCS internal tool that transforms between TCXL and the TC/LINK TCSI format you get by turning on the …\General\Maildebug trace. You can use this to validate a message in TCXL format.

8.4.4 Used XML Library

The library used for XML processing is Microsoft XML 4.0. The necessary files are installed by TC/LINK setup. Any restrictions for Microsoft XML 4.0 also apply to TC/LINK-XML.

8.4.5 MQ Tools

The KCS MQ tools File2MQ.exe and MQ2File.exe have been updated to support the new TC/XML and TCXL message format.

8.4.6 Troubleshooting

Errors are reported in the following way:

- Errors concerning the configuration, resources and failure of the application are reported to the application event log.
- In case of an irrecoverable error or in case of a resource shortage TC/LINK-MQ will shut down.
- Message related errors are reported in a non-delivery notification.

For debugging the recommended trace levels are:

…\General\Tracelevel = 100	Debugging of tclmq.dll and tclink.exe conversion
…\General\Maildebug = 1	Output of Mail (MQ) side TCSI message object
…\Topcall\TCSIdebug = 1	Output of KCS side TCSI message object

In case of problems the following steps are recommended:

- Look for a non-delivery notification.
- Take a look at the event log.
- Take a look at the trace-file. If the recommended trace levels are not set already, set them, restart TC/LINK-MQ and reproduce the problem.

Non-delivery Notifications:

In case of a syntax error in a TCXL message (no XSLT transformation style sheet used) TC/LINK-MQ is able to provide the tag-name or line/column where the error occurred. This is not possible when the XSLT transformation is used. In this case the following is recommended:

- Use the XML-tool msxls.exe and the configured XSLT transformation style sheet (…\Options\In\XMLTransform) to transform the failed XML message to the TCXL format.
- Use the XML-tool XML2tc.exe to transform the TCXL message to the TCSI format.

Either of these two transformations should report an error.

Examples for Non-Deliveries:

Non-Delivery caused by Syntax Error (TC/XML Format):

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<NOTIFICATION xmlns="http://www.topcall.com/XMLSchema/2002/tc/xml">
<NF>NO</NF>
<ARCHIVE>NO</ARCHIVE>
<REMSG>NO</REMSG>
<TO>
<ACTIVE>YES</ACTIVE>
<SERVICE>TOPCALL</SERVICE>
<NUMBER>MS</NUMBER>
<ACTIVE>YES</ACTIVE>
</TO>
<NFINFO>
<STATUS>NONDEL</STATUS>
<TIME>2003-01-28T19:40:13</TIME>
<LACTION>LN</LACTION>
<LNOTE>XML Parsing Error</LNOTE>
</NFINFO>
<TXT> ------------------------------------------
Found an error in transaction file:
XML Parsing Error

Original transaction file is attached.
 ------------------------------------------
</TXT>
<ATT>
<NAME>1</NAME>
</ATT>
</NOTIFICATION>
```

Non-Delivery caused by Syntax Error (TCXL Format):

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<set_entry_ms_mail xmlns="http://www.topcall.com/XMLSchema/2002/tcxl">
   <int_msg_type>10</int_msg_type>
```

```
   <set_entry_ms_mail_orig.set_entry_ms_mail>
      <int_state>420</int_state>
      <time_action>2003-01-28T19:44:17</time_action>
      <ts_last_mda_action>LN</ts_last_mda_action>
      <ts_last_mda_note>XML Parsing Error</ts_last_mda_note>
   </set_entry_ms_mail_orig.set_entry_ms_mail>
   <un_content.l_env_cont>
      <set_header>
         <int_termination>786432</int_termination>
         <l_recipients>
            <set_entry_rs>
               <int_active>1</int_active>
               <int_type>1</int_type>
               <l_full_addr>
                  <set_full_address>
                     <ts_service>TOPCALL</ts_service>
                     <int_active>1</int_active>
                     <un_public_address.set_tc_address>
                        <ts_tc_node></ts_tc_node>
                        <ts_tc_userid>MS</ts_tc_userid>
                     </un_public_address.set_tc_address>
                  </set_full_address>
               </l_full_addr>
            </set_entry_rs>
         </l_recipients>
      </set_header>
      <obj_body_part/>
      <set_att_obj>
         <int_content_type>1076</int_content_type>
         <un_content.blk_binary.tctext> ----------------------------------------
Found an error in transaction file:
XML Parsing Error in l_recipientssdf Line 7, Column 20

Original transaction file is attached.
 ----------------------------------------
</un_content.blk_binary.tctext>
      </set_att_obj>
      <set_att_obj>
         <int_content_type>1024</int_content_type>
         <ts_tos_folder>1</ts_tos_folder>
      </set_att_obj>
   </un_content.l_env_cont>
</set_entry_ms_mail>
```

Non-Delivery caused by Document Conversion Error (TC/XML Format):

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<NOTIFICATION xmlns="http://www.topcall.com/XMLSchema/2002/tc/xml">
<SUBJECT>Delivery Failure: Using TCDCLINK</SUBJECT>
<FROM>
<ACTIVE>YES</ACTIVE>
<SERVICE>FAX</SERVICE>
<NUMBER>8227</NUMBER>
<AB></AB>
<ACTIVE>YES</ACTIVE>
</FROM>
<TO>
<ACTIVE>YES</ACTIVE>
<SNAME>MS</SNAME>
<C1>00000000</C1>
<C2>00000000</C2>
<SERVICE>TOPCALL</SERVICE>
<NUMBER>MS</NUMBER>
<ACTIVE>YES</ACTIVE>
</TO>
<NFINFO>
<STATUS>NONDEL</STATUS>
```

```
<TIME>2003-01-22T16:32:50</TIME>
<LACTION>LG</LACTION>
<LNOTE>Document conversion fail</LNOTE>
<COST>0</COST>
</NFINFO>
<TXT>TOPCALL NON Delivery Notification
---------------------------------------------------------------

Message         : "Using TCDCLINK"  (ID: )
  created by    :

could NOT be sent...
  to Receiver   :  ()
  Reason        : Document conversion failed (LG)
  last Retry at : 22-JAN-2003 16:32:50
  Costs         : 0 for Costcenter:

---------------------------------------------------------------
</TXT>
</NOTIFICATION>
```

Non-Delivery caused by Faxing Error (TC/XML Format):

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<NOTIFICATION xmlns="http://www.topcall.com/XMLSchema/2002/tc/xml">
<SUBJECT>Delivery Failure: A Test Message</SUBJECT>
<FROM>
<ACTIVE>YES</ACTIVE>
<P>NORM</P>
<SERVICE>FAX</SERVICE>
<NUMBER>227</NUMBER>
<AB></AB>
<ACTIVE>YES</ACTIVE>
</FROM>
<TO>
<ACTIVE>YES</ACTIVE>
<P>NORM</P>
<SCOPY>YES</SCOPY>
<HLINE>YES</HLINE>
<RESOLUTION>NORM</RESOLUTION>
<NF>NO</NF>
<ARCHIVE>NO</ARCHIVE>
<REMSG>NO</REMSG>
<EXDATE>2061-01-19</EXDATE>
<EXTIME>03:14:07</EXTIME>
<C1>00AEAD14</C1>
<C2>00000044</C2>
<SERVICE>TCFI</SERVICE>
<NUMBER>MS</NUMBER>
<ACTIVE>YES</ACTIVE>
</TO>
<NFINFO>
<STATUS>NONDEL</STATUS>
<TIME>2003-01-22T17:32:00</TIME>
<DOCNR></DOCNR>
<LACTION>XL</LACTION>
<LNOTE>no fax machine detected</LNOTE>
<MCORR>00011447557</MCORR>
</NFINFO>
<TXT>TOPCALL NON Delivery Notification
---------------------------------------------------------------

Message         : "A Test Message"  (ID: 00011447557)
  created by    :

could NOT be sent...
  to Receiver   :  ()
```

```
  Reason        : no fax machine detected (XL)
  last Retry at : 22-JAN-2003 17:32:00
  Costs         :  for Costcenter: GUEST

------------------------------------------------------------
</TXT>
</NOTIFICATION>
```

Possible problems and solutions:

- Link does not start.
  Event-log entry: ID 8400 “TOM/CTomXml::Enable: CoCreateInstance of pStyleSheetIn failed (Return Code: 80040154)”
  Cause: The Microsoft XML files are missing or not registered.
  Solution: Reinstall the link.
- Link does not start.
  Event-log entry: ID 8400 “TOM/CTomXml::Enable: pStyleSheetIn->load failed; Unable to load XSLT Style Sheet: c:\tcoss\tclp\TCXMLInsdf.xslt (Return Code: 1)”
  Cause: The transformation style sheet is missing or not correct.
  Solution: Set the path correctly or correct the style sheet.
- Link fails after first send-attempt.
  Event-log entry: ID 5102 “wLNK_GetMsg: Internal error 5002 (CoCreateInstance of pWtr failed). The parameters of this log entry and previous event log entries may give more information. Check the trace file (tracelevel 100, maildebug, tcsidebug).”
  Cause: The Microsoft XML files are missing or not registered.
  Solution: Reinstall the link.

8.5 Registry Values Used by TC/LINK-MQ

TC/LINK-MQ uses the following special registry keys:

(Location HKEY_LOCAL_MACHINE\Software\TOPCALL\TCLINKMQ\Options\...)

Registry Key	Type	Default	Possible Values / Meaning
CorrelIDForMQ	REG_SZ		Set this value to 48 character string that corresponds to 24 Byte MQ message correlation ID. This value is used to set correlation ID for all the messages from KCS to MQ. If CorrelIDForMQ and StoreMQCorrelIDInCorr are both set, then ONLY CorrelIDForMQ setting is considered.
OverrideMatchCorrelID	DWORD	0	Defines whether CorrelID match in MQ to KCS message flow should be done or not. When set to 1, no CorrelID match will be done in MQ to KCS message flow. **Note**: This registry values should be configured when you do not require linked message attachments.
DefaultFolder	STRING	+MAIL5V	Defines the default TOS-folder. This is used for: INC: - objects with no defined folder. E.g. entering “FIS” will take all included objects from FIS folder by default.
DefaultOriginatorNumber	STRING	-	Defines the default number for the originator, if it is not defined in the FROM: object of the TCFI message. Must be only the Number or - in case of X400 - the full X400 address
DefaultOriginatorService	STRING	MQ	Defines the default service for the originator, if not defined in the FROM: object of the TCFI message.
DefaultRecipientService	STRING	FAX	Defines the default service for the recipient, if it is not defined in the TO: object of the TCFI message.
DefaultTemplate	STRING	-	Defines the default template, if there no “TEMPLATE=“-parameter in the general part of the TOM-message.

DefaultUserName	STRING	-	Defines the default short name for the originator, if it is not defined in the FROM: object of the TCFI message.
DLQPoll	DWORD	0 (Disabled)	The dead-letter queue shall be polled any <nnn> TCLINK poll cycles. Content is sent to <Operator>
ClusterSupport	DWORD	0	0 No WebSphere MQ Cluster Support 1 WebSphere MQ Cluster Support enabled with OpenOptions MQOO_BIND_ON_OPEN 2 WebSphere MQ Cluster Support enabled with OpenOptions MQOO_BIND_NOT_FIXED; this setting is necessary when using ALIAS queues; Restriction: only XML format is allowed with XMLAttachments = “embedded” For more information, see the section WebSphere MQ Cluster Support.
EnhMemManagment	DWORD	0	0 MQ messages are always fetched with configured maximum messages size 1 Message size is requested from MQ-server and allocated dynamically
Expiry	DWORD	1000	The maximum time until delivery to the IBM WebSphere MQ destination (in seconds). If delivery is not possible within that time; an expiration report will be generated. Setting this key to 0 enables an infinite expiry on MQ. Note: This setting is used only if “at next node” is enabled (…\TOPCALL\NotifMail = 1)!
InterfaceLevel	DWORD	0	Indicates the version of TC Open Message Format to be used. Currently supported: 0 … No extensions. 1 … Level1 extensions (reception time, attachment size, page count) 2 … Level 2 extensions (latest delivery time, keyinfo) 3 … Level 1 and Level 2 extensions
LinkNum	DWORD	0	Unique Link number for current instance of TC/LINK-MQ. This is used to guarantee unique CorrelIds on paralleled links.
MaxActRecp	DWORD (1..100)	1	Indicates the maximum number of active recipients per message from KCS.
MessageSegmentation	DWORD	0	Enable support for MQ message segmentation. If set to 1, messages that are too large for the queue manager or queue to handle are segmented and reassembled by TC/LINK-MQ.
MQTrace	DWORD	0	1 … binary dump of all objects get/put from/to MQ
NoConvert	DWORD	0 (TOM) 1 (XML)	This parameter defines if the transaction file should be converted to the TOPCALL\PCCodepage configured value. Typically TOM format files are in a system depending format and have to be converted, XML files are in UTF-8 and must not be converted, therefore the default values: 1 No conversion of transaction files, default if Options\In\Format = “XML” 0 Conversion of transaction files, default for Options\In\Format <> “XML”
NonDelEventLog	DWORD	0	When set to 1 for each immediate non-delivery notification an event log entry is created. Immediate non-delivery notifications are created by TC/LINK before a message comes to KCS; they are not counted by TC/Monitor.

Operator	STRING	postmaster	KCS User-ID of Operator (to send dead-letter queue content, problem reports, …)
Queue2TC	STRING	2TC.QUEUE	Name of the “To KCS Queue”
QueueDirsync	STRING	DIRSYNC. QUEUE	Name of the “Dirsync Queue”. If this is empty, then Dirsync is disabled.
QueueManager	STRING	TC.QMGR	Name of the Queue Manager to connect to.
QueueNotif	STRING	NOTIF. QUEUE	Name of the “Notification Queue”
QueueTC2	STRING	TC2.QUEUE	Name of the “From KCS Queue”
UseShadowUser	STRING	yes	Kind of addressing (no means no shadow user lookup)
In\TomCodePage	STRING	“PCCodePage”	Coding as requested when pulling the transaction file from IBM Websphere MQ. Possible values are: PCCodePage Windows code page as defined by registry key General\PCCodePage Utf8Bom UTF-8 with byte order mark Utf16LeBom UTF-16 Little Endian with byte order mark
Out\TomCodePage	STRING	“PCCodePage”	Coding of outgoing TOM transaction files, as they are put to IBM Websphere MQ. Possible values: PCCodePage Windows code page as defined by registry key General\PCCodePage Utf8Bom UTF-8 with byte order mark Utf16LeBom UTF-16 Little Endian with byte order mark
StoreMQCorrelIDInCorr	DWORD	0	set this value to any of 0,1,2,3,4,5 0 – Default value, disable this functionality. 1/2/3/4/5 – Corresponds to TS_CORREL_x field in KCS messages. When set, this value is used to set the correlID of MQ message in the corresponding recipient’s TS_CORREL_x field of KCS messages. When a subsequent notification from the KCS arrives to TCLINKMQ, the stored TS_CORREL_x value is used as MQ notification message’s correlID. If CorrelIDForMQ and StoreMQCorrelIDInCorr are both set, then ONLY CorrelIDForMQ setting is considered.

Note:

- TC/LINK-MQ must be stopped and restarted before changes become effective.
- For common Registry entries, refer to the TCLINK Manual!

8.5.1 Special Registry Keys for XML Format

The following registry keys are used to configure the TC/LINK-MQ format. The “In” parameters are for inbound messages (Mail to KCS), the “Out” parameters are for outbound messages (KCS to Mail).

HKEY_LOCAL_MACHINE\Software\TOPCALL\TCLINKMQ\Options\

Registry Key	Type	Default Value	Description
In\Format	STRING	“TOM”	The format used for incoming messages (Mail to KCS); either “TOM”, “XML” or “NON” for transparent mode.
Out\Format	STRING	“TOM”	The format used for outgoing messages (KCS to Mail); either “TOM”, “XML” or “NON” for transparent mode.
In\XMLTransform	STRING	“C:\TCOSS\TCLP\TCXMLIn.xlst”	Name and path of the transformation style sheet for incoming messages. Two values (style sheets) are possible, separated by “;” (see also 8.5.2). The result of this transformation has to be TCXL. If empty, the transformation is skipped.

Out\XMLTransform	STRING	"C:\TCOSS\TCLP\TCXMLOut.xlst"	Name and path of the transformation style sheet for outgoing messages. Two values (style sheets) are possible, separated by ";" (see also 8.5.2). The input of this transformation is TCXL. If empty, the transformation is skipped.
Out\XMLEncoding	STRING	"UTF-8"	Outbound XML encoding (only used when there is no Style Sheet conversion, else the style sheet defines the encoding). Valid and tested values: "UTF-8", "UTF-16", "ISO-8859-1".
Out\XMLAttachments	STRING	"linked"	Defines if attachments of outgoing messages are only referenced (similar to the classic TOM format) or inside the XML message; either "linked" or "embedded".
Out\OverrideXmlOutCodePage	DWORD	0	When set to 1, the CCSID is set to the value of TOPCALL\TCLINKMQ\General\PCCodePage Otherwise, the CCSID is set to 1208.

The "In\Format" and "Out\Format" keys define if the classic TOM format is used or the new TC/XML format. Accordingly the other keys are ignored if the classic TOM format is configured.

8.5.2 Double XSL Conversion Support

Two XML conversion steps are possible. That makes it easier to create customized conversion style sheets, as the conversion can be done from and to the easier TC/XML format rather than the more complicated TCXL format.

The transformation is done from left to right, that means that first the left style sheet is used, then the right one. See the following examples:

- In\XMLTransform: "c:\tcoss\tclp\xmltools\CustomIn.xslt;c:\tcoss\tclp\TCXMLIn.xslt"

"CustomIn.xslt" transforms from the custom format to TC/XML, afterwards "TCXMLIn.xslt" transforms from TC/XML to TCXL.

- Out\XMLTransform: "c:\tcoss\tclp\TCXMLOut.xslt;c:\tcoss\tclp\xmltools\CustomOut.xslt"

"TCXMLOut.xslt" transforms from TCXL to TC/XML, afterwards "CustomOut.xslt" transforms from TC/XML to the custom format.

8.6 Transparent Mode

In order to allow customized Link-Exits to implement their own message format, it is possible to switch TC/LINK-FI and TC/LINK-MQ to a transparent mode where the message content is ignored.

For incoming messages (to TOPALL) this means that a transaction file is taken and put as it is to the text part of the TC/LINK message. Some defaults are set for this message, but a special Link-Exit is supposed to parse the text message and complete all necessary message parameters.

For outgoing messages (from KCS) only the text part of the TC/LINK message is written to the transaction file, all other message parameters are ignored. Here the TC/LINK-Exit is supposed to put all necessary information to the text part of the message.

When TOPCALL\TCLINKMQ\Options\In\Format is set to 'NON' (transparent mode), transaction message content is stored in TCCodePage format instead of Unicode format in TCOSS.

8.6.1 Defaults for Incoming Messages

For incoming messages (Mail to KCS) default values are set like for the TOM and XML format. The following TCSI parameters are set:

SET_HEADER Object:

Object	Default	Description
INT_TERMINATION	NOTIF_NEG	Flags if Notifications for Non-Delivery and Delivery have to be returned, the default is for Non-Delivery only

SET_ENTRY_RS_ ORIGINATOR	Taken from registry	Originator of the message, default is taken from the registry (…TCLINKFI\Options\DefaultOriginatorService and …DefaultOriginatorNumber)

8.6.2 Configuration

To configure the new transparent mode use the HKEY_LOCAL_MACHINE\Software\TOPCALL\TCLINKFI\Options\In\Format and …Out\Format registry keys with the value “NON”.

8.7 Samples for IBM WebSphere MQ Programming

Please note that the samples below are given for demonstration purposes only; therefore, without any warranties! Especially, you need to do the following:

- Make all hard-coded parameters configurable
- Add functionality to handle multiple attachments in single message
- Adapt message generation/retrieval to fit your connected applications
- Design your own way to ensure unique CorrelIds
- Improve error handling
- Test the code in your special environment (operating system, IBM WebSphere MQ releases, …)

Before you can actually run the samples, you need to install the IBM WebSphere MQ client on your machine, and define the connection to the IBM WebSphere MQ server on the TC/LINK-MQ server. See the IBM WebSphere MQ client manual for details!

Example:

(Assuming that TC/LINK-MQ, and the IBM WebSphere MQ server are both located at IP address 193.81.166.46; queue manager default configuration from MQSetup.txt)

- Enter <MQSERVER=TC.CHANNEL/TCP/193.81.166.46> as a user variable on Windows (Control Panel/System/Environment).
- Type <export MQSERVER=TC.CHANNEL/TCP/193.81.166.46> on UNIX systems.

8.7.1 Compilation Hints

8.7.1.1 Microsoft Developer Studio 5.0 (Tested on Windows NT 4.0)

Create a new project (“Windows Console Application”), insert the appropriate source file to the project, and make sure to link the right MQ library (MQIC32.lib for client applications, MQM.lib for server)

The source code should compile without any problems.

8.7.1.2 GNU C/C++ 2.8 (Tested on HP-UX 10.20)

To compile the samples, simply type (from the path where the source file is located)

```
<gcc -o mq2tc mq2tv.c –lmqic>
```

Note:

- GNU “binutils” (2.08 or higher) must be installed! The native HP-UX 10.20 assembler will fail for assembling the GNU output.
- The “-lmqic” indicates linkage to the IBM WebSphere MQ client library. If you want to connect to the local server, use “-lmqm” instead.

8.7.2 MQ2TC.C - A Simple Client for Sending to KCS (ANSI C)

(The source file is available on the TC/LINK-MQ server in the directory “c:\tcoss\tclink\mqtools”)

```
/**************************************************************************
  This is a sample program that writes a simple message plus a single
  attachment to IBM WebSphere MQ.

  All parameters are hard-coded for easiest sample code:
```

```
    * queue manager name
    * queue name to KCS
    * message text
    * attachment data
    * message expiry interval
*****************************************************************************/

#include <stdio.h>
#include <string.h>
#include "cmqc.h"


/* My queue manager */
#define QUEUE_MANAGER    "TC.QUEUE.MANAGER"
/* My queue to KCS */
#define QUEUE_2TC        "2TC.QUEUE"
/* Maximum delivery time: 10 minutes (is in 1/10 sec) */
#define MYEXPIRY         (6000)

/* Here comes the sample message text */
#define MYTESTMESSAGE    "SUBJECT=Testmessage from MQ2TC sample\n" \
                         "FROM: SERVICE=MQ, NUMBER=Administrator\n" \
                         "TO: SERVICE=TOPCALL, NUMBER=postmaster\n" \
                         "TXT:\n" \
                         "Welcome to TC/LINK-MQ!\n" \
                         ":ATT: NAME=1, APPLICATION=Att.bin, COMMENT=binary attachment"

/* And this is a (pseudo-) binary attachment */
#define MYATTACHMENT    "This attachment is sent as binary data."


/*****************************************************************************
  Build a unique correlation identifier
*/
MQLONG MakeUniqueId (MQBYTE24 id,                /* Field to put the unique id */
                     MQLONG attnum)              /* desired attachment number */
{
    static int i=0;
    /* TODO adapt the algorithm to fit your environment! */
    sprintf ((char*)id, "ABCDEFGHIJ%10.10i", i);

    /* put attnum to last four bytes (I restrict it to 1 byte here) */
    id[20] = id[21] = id[22] = 0;
    id[23] = (char)attnum;
    return MQRC_NONE;
}

/*****************************************************************************
  Write a Message to IBM WebSphere MQ
  Return reason code.
*/
MQLONG PutMQMsg (MQLONG Hcon,                    /* Connection handle */
                 MQLONG Hobj)                    /* Queue object handle */
{
    MQLONG CompCode, Reason;                     /* completion/reason code */
    MQMD     md = {MQMD_DEFAULT};                /* Message Descriptor */
    MQPMO    pmo = {MQPMO_DEFAULT};              /* put message options */

    /*************************************************************************/
    /* first put attachment file */
    MakeUniqueId (md.CorrelId, 1);
    md.Expiry = MYEXPIRY;

    /* Some options */
    pmo.Options = MQPMO_SYNCPOINT +              /* single-phase commit used */
        MQPMO_NEW_MSG_ID +                       /* generate new MsgId */
```

```
        MQPMO_FAIL_IF_QUIESCING;                /* stop if QMGR shutting down */

    MQPUT(Hcon,                                 /* connection handle */
        Hobj,                                   /* queue object handle */
        &md,                                    /* message descriptor */
        &pmo,                                   /* default options (datagram) */
        sizeof(MYATTACHMENT),                   /* buffer length */
        MYATTACHMENT,                           /* message buffer */
        &CompCode,                              /* completion code */
        &Reason);                               /* reason code */

    if (Reason != MQRC_NONE)
    {
        printf("MQPUT (attachment) ended with reason code %ld\n", Reason);
        return Reason;
    }

    /*********************************************************************/
    /* now add the transaction message */
    /* Need same CorrelId as for attachment, but ending with 0 */
    md.CorrelId[20] = md.CorrelId[21] = md.CorrelId[22] = md.CorrelId[23] = 0;
    md.Expiry = MYEXPIRY;
    /* transaction file needs character string format */
    memcpy(md.Format, MQFMT_STRING, (size_t)MQ_FORMAT_LENGTH);

    /* Some options */
    pmo.Options = MQPMO_SYNCPOINT +             /* single-phase commit used */
        MQPMO_NEW_MSG_ID +                      /* generate new MsgId */
        MQPMO_FAIL_IF_QUIESCING;                /* stop if QMGR shutting down */

    MQPUT(Hcon,                                 /* connection handle */
        Hobj,                                   /* queue object handle */
        &md,                                    /* message descriptor */
        &pmo,                                   /* default options (datagram) */
        sizeof(MYTESTMESSAGE),                  /* buffer length */
        MYTESTMESSAGE,                          /* message buffer */
        &CompCode,                              /* completion code */
        &Reason);                               /* reason code */

    if (Reason != MQRC_NONE)
    {
        printf("MQPUT (transcation message) ended with reason code %ld\n", Reason);
        return Reason;
    }

    return MQRC_NONE;
}

/*****************************************************************************
  Main function call - no commandline parameters.
*/
int main (int argc,                             /* argument count */
          char *argv[])                         /* argument list */
{
    MQLONG CompCode, OpenCode, Reason;          /* completion/reason code */
    MQHCONN  Hcon;                              /* connection handle */
    MQOD     od = {MQOD_DEFAULT};               /* Object Descriptor */
    MQHOBJ   Hobj;                              /* object handle */
    MQLONG   Options;                           /* MQOPEN/MQCLOSE options */

    printf ("This is a C-sample to send a single message to KCS\n");

    /* connect to queue manager */
    MQCONN(QUEUE_MANAGER,                       /* queue manager name */
        &Hcon,                                  /* connection handle */
        &CompCode,                              /* completion code */
```

```
        &Reason);                                   /* reason code */

    if (CompCode == MQCC_FAILED)
    {
      printf("MQCONN ended with reason code %ld\n", Reason);
      return Reason;
    }

     /* Open the message queue for output */
     strcpy(od.ObjectName, QUEUE_2TC);
     Options = MQOO_OUTPUT                          /* we want to write messages! */
         + MQOO_FAIL_IF_QUIESCING;                  /* but not if QM stopping */

     MQOPEN(Hcon,                                   /* connection handle */
         &od,                                       /* object descriptor for queue */
         Options,                                   /* open options */
         &Hobj,                                     /* object handle */
         &OpenCode,                                 /* completion code */
         &Reason);                                  /* reason code */

     if (Reason != MQRC_NONE || OpenCode == MQCC_FAILED)
     {
         printf("unable to open queue for output (reason %ld)\n");
     }
     else
     {
         /* Put the message */
         Reason = PutMQMsg (Hcon, Hobj);
         if (Reason == MQRC_NONE)
         {
             /* ok -> commit  */
             MQCMIT (Hcon,                          /* connection handle */
                 &CompCode,                         /* Completion code */
                 &Reason);                          /* Reason code qualifying CompCode */
         }
         else
         {
             /* failed -> forget the message! */
             MQBACK (Hcon,                          /* connection handle */
                 &CompCode,                         /* Completion code */
                 &Reason);                          /* Reason code qualifying CompCode */
         }

         if (Reason != MQRC_NONE)
         {
             printf("MQCMIT/MQBACK failed with reason %ld\n", Reason);
         }
     }

     /* Close the message queue (if open was successful */
     if (OpenCode != MQCC_FAILED)
     {
         MQCLOSE(Hcon,                              /* connection handle */
             &Hobj,                                 /* object handle */
             0,                                     /* no options required here */
             &CompCode,                             /* completion code */
             &Reason);                              /* reason code */

         if (Reason != MQRC_NONE)
         {
             printf("MQCLOSE ended with reason code %ld\n", Reason);
         }
     }

     /* Disconnect from MQM */
     MQDISC(&Hcon,                                  /* connection handle */
```

```
        &CompCode,                                  /* completion code */
        &Reason);                                   /* reason code */

    if (Reason != MQRC_NONE)
    {
        printf("MQDISC ended with reason code %ld\n", Reason);
    }

    printf("Thanks for using TO/LINK-MQ!\n");
    return Reason;
}
```

8.7.3 TC2MQ.C - A Simple Client for Reception from KCS (ANSI C)

(The source file is available on the TC/LINK-MQ server in the directory “c:\tcoss\tclink\mqtools”)

```
/*****************************************************************************
  This is a sample program that fetches a simple message (with or without
  attachment) from IBM WebSphere MQ.

  All parameters are hardcoded for easiest sample code:
    * queue manager name
    * queue name to KCS
*****************************************************************************/

#include <stdio.h>
#include <malloc.h>
#include <string.h>
#include "cmqc.h"

/* My queue manager */
#define QUEUE_MANAGER    "TC.QUEUE.MANAGER"
/* My queue to KCS */
#define QUEUE_TC2        "TC2.QUEUE"
/* property identifying a transaction message */
#define TFIDX            (0)
/* Syntax for attachments */
#define ATTID            ":ATT: NAME="
/* Desired character set */
#define CCSID            (850)

/*****************************************************************************
  Search given queue object for any MQ messages, get first one that matches
  Returns the length of found message (-1 on error);
*/
MQLONG BrowseMsg (MQHCONN Hcon,                 /* Connection handle */
                  MQHOBJ Hobj,                  /* Queue Object Handle */
                  MQMD md,                      /* Message descriptor */
                  MQLONG idx)
{
    MQMD  DefMd = {MQMD_DEFAULT};               /* default message descriptor */
    MQGMO   gmo = {MQGMO_DEFAULT};              /* get message options */
    MQLONG MsgLen, CompCode, Reason;
    MQBYTE24 Id;

    /* store CorrelId for later comparison */
    memcpy (Id, md.CorrelId, sizeof(Id));

    /* set initial options */
    gmo.Options = MQGMO_NO_WAIT                 /* don't wait for new messages */
        + MQGMO_BROWSE_FIRST                    /* set browse cursor to start of queue
*/
        + MQGMO_ACCEPT_TRUNCATED_MSG            /* as we don't give a buffer in the
first run */
        + MQGMO_CONVERT;                        /* convert (as we search transaction
file first) */
```

```
    /* Loop until there is a failure, or a MQ message found */
    while (CompCode != MQCC_FAILED)
    {
        /* Reset message descriptor */
        memcpy(&md, &DefMd, sizeof(DefMd));

        /* Browse for message */
        MQGET(Hcon,                                 /* connection handle */
            Hobj,                                   /* object handle */
            &md,                                    /* message descriptor */
            &gmo,                                   /* get message options */
            0,                                      /* buffer length */
            NULL,                                   /* message buffer */
            &MsgLen,                                /* message length */
            &CompCode,                              /* completion code */
            &Reason);                               /* reason code */

        if (Reason == MQRC_TRUNCATED_MSG_ACCEPTED ||
            Reason == MQRC_NONE)
        {
            /* now check if it matches given parameters */
            if (idx == 0 && md.CorrelId[20] == 0 && md.CorrelId[21] == 0 &&
                        md.CorrelId[22] == 0 && md.CorrelId[23] == 0)
            {
                printf ("Found transaction message\n");
                return MsgLen;
            }

            if (idx != 0 && idx == md.CorrelId[23] &&
                0 == memcmp (md.CorrelId, Id, 20))
            {
                printf ("Found attachment message\n");
                return MsgLen;
            }
        }
        /* prepare options for next run */
        gmo.Options = MQGMO_NO_WAIT             /* don't wait for new messages */
            + MQGMO_BROWSE_NEXT                 /* move browse cursor */
            + MQGMO_ACCEPT_TRUNCATED_MSG        /* as we don't give a buffer */
            + MQGMO_CONVERT;                    /* convert if necessary */
    }

    printf ("MQGET (browse) returned %ld\n", Reason);
    return -1;
}

/**************************************************************************
  Search given queue object for transaction files, get first one.

*/
MQLONG GetMQMsg (MQHCONN Hcon,                  /* Connection handle */
                 MQHOBJ Hobj)                   /* Queue Object Handle */
{
    MQMD     md = {MQMD_DEFAULT};               /* Message Descriptor */
    MQGMO   gmo = {MQGMO_DEFAULT};              /* get message options */
    PMQBYTE pData, pAtt;
    MQLONG CompCode, Reason, MsgLen, AttIdx;
    PMQCHAR pAttId;

    /* Position browse cursor to transaction message */
    MsgLen = BrowseMsg (Hcon,                   /* Connection handle */
        Hobj,                                   /* Queue Object Handle */
        md,                                     /* Message descriptor */
        TFIDX);                                 /* Transaction file index */

    if (MsgLen >= 0)
```

```
    {
        /* set some important message options */
        gmo.Options = MQGMO_NO_WAIT              /* don't wait for new messages */
            + MQGMO_MSG_UNDER_CURSOR             /* get message under cursor */
            + MQGMO_SYNCPOINT                    /* it's mine now */
            + MQGMO_CONVERT;                     /* convert (as we search transaction
file first) */

        /* allocate buffer */
        pData = malloc (MsgLen+1);
        if (pData == 0)
        {
            printf ("memory allocation failed\n");
            return MQRC_RESOURCE_PROBLEM;
        }

        /* get the message */
        md.CodedCharSetId = CCSID;
        MQGET(Hcon,                              /* connection handle */
            Hobj,                                /* object handle */
            &md,                                 /* message descriptor */
            &gmo,                                /* get message options */
            MsgLen,                              /* buffer length */
            pData,                               /* message buffer */
            &MsgLen,                             /* message length */
            &CompCode,                           /* completion code */
            &Reason);                            /* reason code */

        if (CompCode != MQCC_OK)
        {
            printf ("MQGET (transcation message) returned %ld\n", Reason);
            free (pData);
            return Reason;
        }

        if (md.MsgType != MQMT_DATAGRAM)
        {
            /* TODO handle report messages or similar! */
            printf("Unexpected message type %ld - deleting\n", md.MsgType);
            free (pData);
            return MQRC_NONE;
        }

        /* output to screen - you should do something more useful here */
        pData[MsgLen] = 0;
        printf (pData);

        /* holding attachments? */
        pAttId = strstr (pData, ATTID);
        if (pAttId)
        {
            /* get the attachment! TODO: loop for multiple attachments */
            AttIdx = atoi (pAttId + strlen(ATTID));
            /* Position browse cursor to transaction message */
            MsgLen = BrowseMsg (Hcon,            /* Connection handle */
                Hobj,                            /* Queue Object Handle */
                md,                              /* Message descriptor */
                AttIdx);                         /* Transaction file index */

            if (MsgLen >= 0)
            {
                /* set some important message options; NOTE: NO MQGMO_CONVERT! */
                gmo.Options = MQGMO_NO_WAIT      /* don't wait for new messages */
                    + MQGMO_MSG_UNDER_CURSOR     /* get message under cursor */
                    + MQGMO_SYNCPOINT;           /* it's mine now */
```

```
                /* allocate buffer */
                pAtt = malloc (MsgLen+1);
                if (pAtt == 0)
                {
                    printf ("memory allocation failed\n");
                    free (pData);
                    return MQRC_RESOURCE_PROBLEM;
                }

                /* get the attachment */
                MQGET(Hcon,                     /* connection handle */
                    Hobj,                       /* object handle */
                    &md,                        /* message descriptor */
                    &gmo,                       /* get message options */
                    MsgLen,                     /* buffer length */
                    pAtt,                       /* attachment buffer */
                    &MsgLen,                    /* message length */
                    &CompCode,                  /* completion code */
                    &Reason);                   /* reason code */

                if (CompCode != MQCC_OK)
                {
                    printf ("MQGET (transcation message) returned %ld\n", Reason);
                    free (pData);
                    free (pAtt);
                    return Reason;
                }
            }

            /* Now got the attachment ... print it */
            pAtt[MsgLen] = 0;
            printf ("Attachment data (first 100 bytes):\n%100.100s\n", pAtt);
            free (pAtt);
        }/* if attachment found */
        free (pData);
    } /* if MsgLen > 0 */
    return MQRC_NONE;
}

/******************************************************************************
  Main function call - no commandline parameters.
*/
int main (int argc, char *argv[])
{
    MQLONG CompCode,OpenCode, Reason;           /* completion codes/reason code */
    MQHCONN  Hcon;                              /* connection handle */
    MQOD     od = {MQOD_DEFAULT};               /* Object Descriptor */
    MQHOBJ   Hobj;                              /* object handle */
    MQLONG   Options;                           /* MQOPEN/MQCLOSE options */

    printf ("This is a C-sample for fetching messages from KCS\n");

    /* connect to queue manager */
    MQCONN(QUEUE_MANAGER,                       /* queue manager name */
        &Hcon,                                  /* connection handle */
        &CompCode,                              /* completion code */
        &Reason);                               /* reason code */

   if (CompCode == MQCC_FAILED)
   {
     printf("MQCONN ended with reason code %ld\n", Reason);
     return Reason;
   }

    /* Open the message queue for output */
    strcpy(od.ObjectName, QUEUE_TC2);
```

```
    Options = MQOO_INPUT_SHARED                 /* we want to read messages! */
        + MQOO_BROWSE                           /* open queue for browsing first */
        + MQOO_FAIL_IF_QUIESCING;               /* but not if MQM stopping */

    MQOPEN(Hcon,                                /* connection handle */
        &od,                                    /* object descriptor for queue */
        Options,                                /* open options */
        &Hobj,                                  /* object handle */
        &OpenCode,                              /* completion code */
        &Reason);                               /* reason code */

    if (Reason != MQRC_NONE || OpenCode == MQCC_FAILED)
    {
        printf("unable to open queue for output (reason %ld)\n");
    }
    else
    {
        /* Put the message */
        Reason = GetMQMsg (Hcon, Hobj);
        if (Reason == MQRC_NONE)
        {
            /* ok -> commit  */
            MQCMIT (Hcon,                       /* connection handle */
                &CompCode,                      /* Completion code */
                &Reason);                       /* Reason code qualifying CompCode */
        }
        else
        {
            /* failed -> forget the message! */
            MQBACK (Hcon,                       /* connection handle */
                &CompCode,                      /* Completion code */
                &Reason);                       /* Reason code qualifying CompCode */
        }

        if (Reason != MQRC_NONE)
        {
            printf("MQCMIT/MQBACK failed with reason %ld\n", Reason);
        }
    }

    /* Close the message queue (if open was successful */
    if (OpenCode != MQCC_FAILED)
    {
        MQCLOSE(Hcon,                           /* connection handle */
            &Hobj,                              /* object handle */
            0,                                  /* no options required here */
            &CompCode,                          /* completion code */
            &Reason);                           /* reason code */

        if (Reason != MQRC_NONE)
        {
            printf("MQCLOSE ended with reason code %ld\n", Reason);
        }
    }

    /* Disconnect from MQM */
    MQDISC(&Hcon,                               /* connection handle */
        &CompCode,                              /* completion code */
        &Reason);                               /* reason code */

    if (Reason != MQRC_NONE)
    {
        printf("MQDISC ended with reason code %ld\n", Reason);
    }

    printf("Thanks for using TC/LINK-MQ!\n");
```

```
    return Reason;
}
```

8.8 Default Configuration File “MQSetup.txt”

This configuration file gives the simplest way of TC/LINK-MQ usage:

- All queues from/to KCS are on the local IBM WebSphere MQ server
- A single server channel on TCP is defined to allow client connections from outside.

```
*******************************************************/
*  Command syntax:                                    */
*     runmqsc qmgrname < MQsetup.txt                  */
*                                                     */
*                                                     */
*   This is the definition file for the KCS        */
*   SERVER environment.                               */
*   All maximum message sizes are set to 4MB in this  */
*   sample file. Edit it according to your needs.     */
*                                                     */
*******************************************************/

*******************************************************/
* LOCAL QUEUE DEFINITIONS                             */
*******************************************************/
    DEFINE QLOCAL('2TC.QUEUE') +
     DESCR('Message Queue from MQ to KCS') +
     PUT(ENABLED) +
     DEFPRTY(0) +
     DEFPSIST(YES) +
     MAXMSGL(4000000) +
     GET(ENABLED)

    DEFINE QLOCAL('TC2.QUEUE') +
     DESCR('Message Queue from KCS to MQ') +
     PUT(ENABLED) +
     DEFPRTY(0) +
     DEFPSIST(YES) +
     MAXMSGL(4000000) +
     GET(ENABLED)

    DEFINE QLOCAL('NOTIF.QUEUE') +
     DESCR('Notification Queue from KCS to MQ') +
     PUT(ENABLED) +
     DEFPRTY(0) +
     DEFPSIST(YES) +
     MAXMSGL(4000000) +
     GET(ENABLED)

    DEFINE QLOCAL('DIRSYNC.QUEUE') +
     DESCR('Directory Synchronization Queue from MQ to KCS') +
     PUT(ENABLED) +
     DEFPRTY(0) +
     DEFPSIST(YES) +
     MAXMSGL(4000000) +
     GET(ENABLED)

*******************************************************/
* A SINGLE SERVER/CLIENT CHANNEL DEFINITION PAIR TO    */
* ALLOW CLIENTS FROM OUTSIDE TO CONNECT TO MY QUEUES   */
*******************************************************/
    DEFINE CHANNEL (TC.CHANNEL) +
           CHLTYPE(SVRCONN) +
           TRPTYPE(TCP) +
           DESCR('Server channel for client connections to TC Queues')
```

```
    DEFINE CHANNEL (TC.CHANNEL) +
           CHLTYPE(CLNTCONN) +
           TRPTYPE(TCP) +
           QMNAME(TC.QUEUE.MANAGER) +
           CONNAME(TOPCALL) +
           DESCR('Client Connection to TC Queues')

*******************************************************/
* Some definitions that are usually already there.     */
* However, by default, recreate them if deleted!       */
*******************************************************/
    DEFINE QLOCAL('SYSTEM.DEAD.LETTER.QUEUE') +
     DESCR('IBM WebSphere MQ default dead queue')

    ALTER QMGR DEADQ(SYSTEM.DEAD.LETTER.QUEUE)

*******************************************************/
* Disable security features of Websphere 7.1 and 8.0   */
*******************************************************/

    ALTER QMGR CHLAUTH(DISABLED)
    ALTER QMGR CONNAUTH('')
```

Note:

- This file is not updated automatically if you change anything during TC/LINK-MQ “Advanced Installation”! You need to edit this manually!
- Automatic creation of MQ dependencies only works with a local IBM WebSphere MQ server (not with the IBM WebSphere MQ client).

8.9 Advanced Configuration for Server-Server Communication

This section gives a configuration file for the scenario described in 2.5 “Connecting to Other IBM WebSphere MQ Servers”. For easier understanding, only one channel pair is used (no separate channels for the NOTIF and DIRSYNC queues).

Note:

- You need to adapt the CONNAME parameters to match the real IP hostnames
- You need to adapt the RQMNAME parameters according to the real queue manager names.
- If you use different IP ports (e.g. due to multiple queue managers sitting on a host), you need to add the correct port number also.

8.9.1 IBM WebSphere MQ Configuration on the TC/LINK-MQ Server

```
*******************************************************/
*  Command syntax:                                    */
*     runmqsc qmgrname < MQsetup.txt                  */
*                                                     */
*  Sample file for Server-server communication        */
*  Local queue manager                                */
*******************************************************/

*******************************************************/
* LOCAL QUEUE DEFINITIONS                             */
*******************************************************/
    DEFINE QLOCAL('2TC.QUEUE') +
     DESCR('Message Queue from Host to KCS') +
     PUT(ENABLED) +
     DEFPRTY(0) +
     DEFPSIST(YES) +
     MAXMSGL(4000000) +
     GET(ENABLED)

    DEFINE QALIAS('DIRSYNC.QUEUE') +
```

```
      TARGQ(2TC.QUEUE) +
     DEFPSIST(YES)

*********************************************************/
* REMOTE/TRANSMISSION QUEUES                            */
*********************************************************/
    DEFINE QREMOTE('TC2.RQUEUE') +
       DESCR('Remote Queue to Host') +
       XMITQ(2UNIX.XMITQ) +
       RNAME(2UNIX.QUEUE) +
       RQMNAME(UNIX.QMGR) +
       DEFPSIST(YES)

    DEFINE QLOCAL('2UNIX.XMITQ') +
     DESCR('Transmission Queue to Host') +
      USAGE (XMITQ)

    DEFINE QALIAS('NOTIF.QUEUE') +
      TARGQ(TC2.RQUEUE) +
     DEFPSIST(YES)

*********************************************************/
* CHANNELS TO/FROM HOST SYSTEM                          */
*********************************************************/
    DEFINE CHANNEL('2UNIX.CHANNEL') +
      CHLTYPE (SDR) +
      TRPTYPE (TCP) +
      CONNAME (unix_hostname) +
      XMITQ (2UNIX.XMITQ)

    DEFINE CHANNEL('2TC.CHANNEL') +
      CHLTYPE (RCVR) +
      TRPTYPE (TCP)
```

8.9.2 IBM WebSphere MQ Configuration on the Remote MQ Server

```
*********************************************************/
*  Command syntax:                                      */
*     runmqsc qmgrname < MQsetup.txt                    */
*                                                       */
*  Sample file for Server-server communication          */
*  Distant queue manager                                */
*********************************************************/

*********************************************************/
* LOCAL QUEUE DEFINITIONS                               */
*********************************************************/
    DEFINE QLOCAL('2UNIX.QUEUE') +
     DESCR('Message Queue from KCS to host') +
     PUT(ENABLED) +
     DEFPRTY(0) +
     DEFPSIST(YES) +
     MAXMSGL(4000000) +
     GET(ENABLED)

*********************************************************/
* REMOTE/TRANSMISSION QUEUES                            */
*********************************************************/
    DEFINE QREMOTE('2TC.RQUEUE') +
       DESCR('Remote Queue to KCS') +
       XMITQ(2TC.XMITQ) +
       RNAME(2TC.QUEUE) +
       RQMNAME(TOPCALL.QMGR) +
       DEFPSIST(YES)

    DEFINE QLOCAL('2TC.XMITQ') +
```

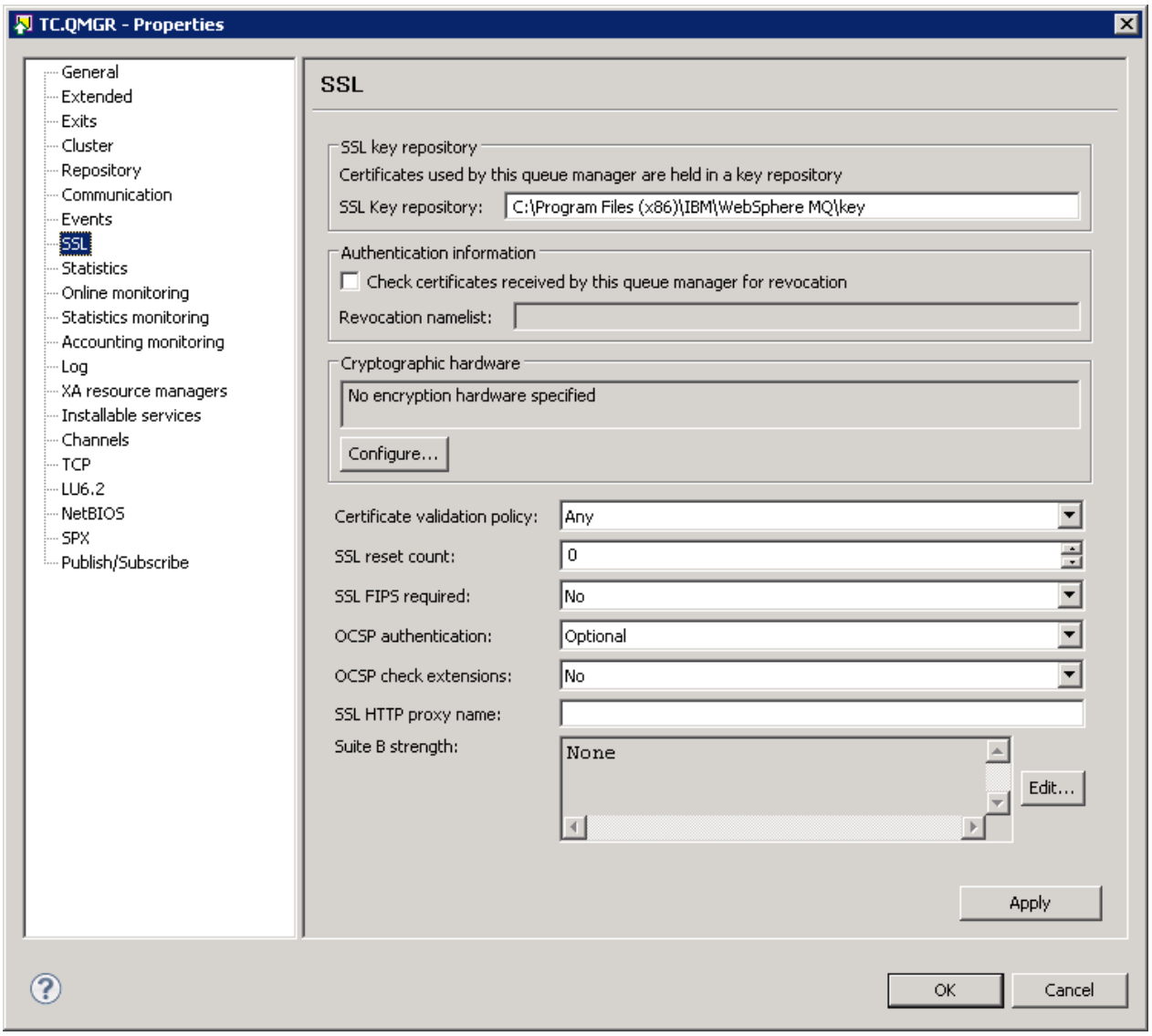
```
    DESCR('Transmission Queue to KCS') +
      USAGE (XMITQ)

*********************************************************/
* CHANNELS TO/FROM HOST SYSTEM                           */
*********************************************************/
    DEFINE CHANNEL('2TC.CHANNEL') +
      CHLTYPE (SDR) +
      TRPTYPE (TCP) +
      CONNAME (topcall_hostname) +
      XMITQ (2TC.XMITQ)

    DEFINE CHANNEL('2UNIX.CHANNEL') +
      CHLTYPE (RCVR) +
      TRPTYPE (TCP)
```

8.10 TC/LINK-MQ with SSL

This example configuration shows how to set up SSL for the MQ Server 7.5 by using a self-signed certificate.

8.10.1 Configuring the IBM WebSphere MQ Server

In the IBM WebSphere MQ Explorer, open the queue manager properties of the TC/LINK-MQ queue manager and select the SSL tab:



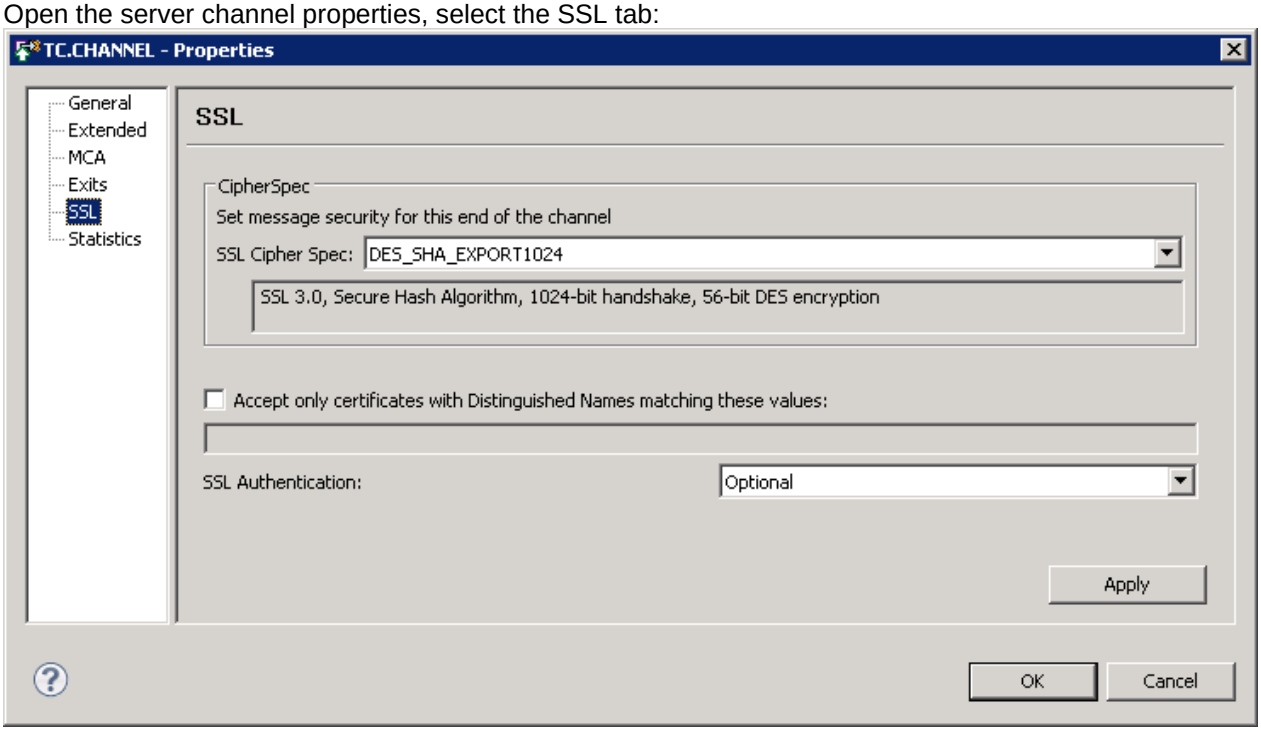
Specify the SSL Key repository without file extension, e.g.:

```
C:\Program Files (x86)\IBM\WebSphere MQ\key
```

Set “OCSP authentication” to Optional.

Set the other parameters as shown.

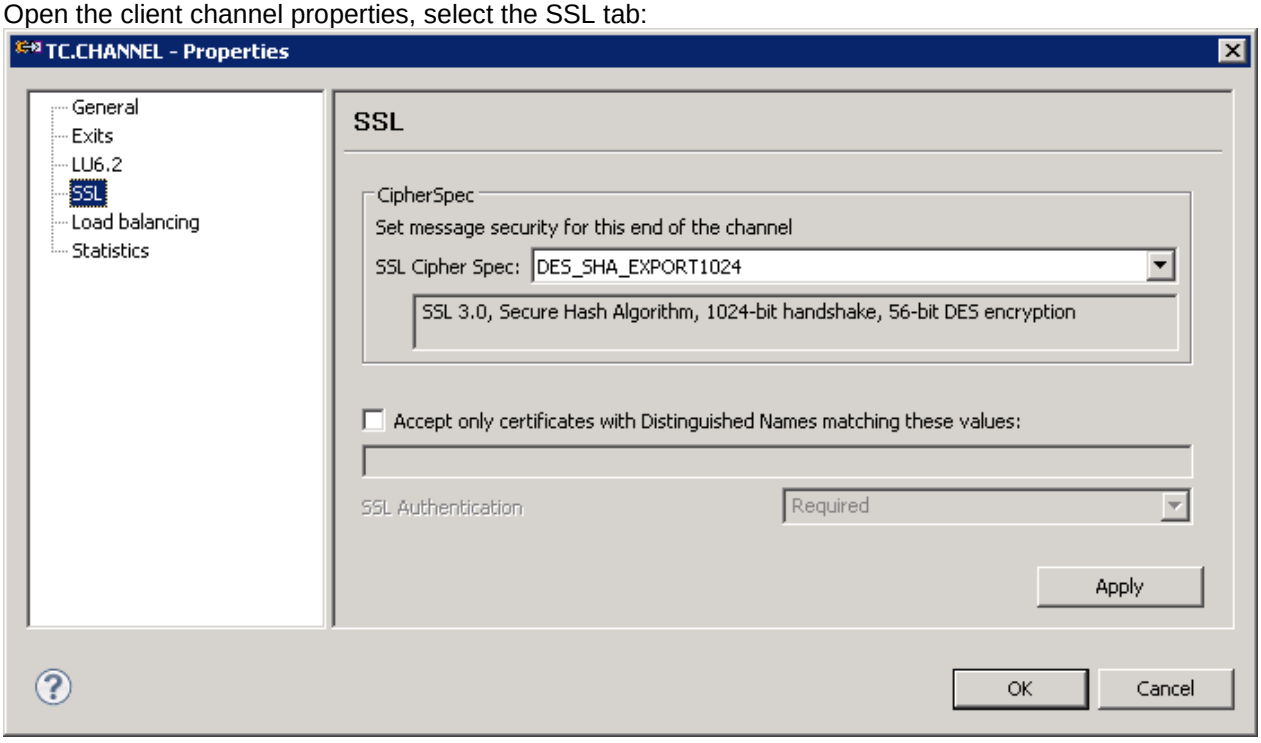
Open the server channel properties, select the SSL tab:



Set one of the SSL Cipher Spec values, e.g. DES_SHA_EXPORT1024.

Set SSL Authentication to Optional (so the client has not to authenticate itself).

Open the client channel properties, select the SSL tab:



Set the same SSL Cipher Spec value as for the server channel.

8.10.2 Configuring the TC/LINK-MQ Computer

Copy the updated MQ queue manager configuration file AMQCLCHL.TAB from the MQ server to the TC/LINK-MQ machine.

Typically the path of the AMQCLCHL.TAB file on the server will be (Websphere MQ 7.5 and before):

```
C:\Program Files (x86)\IBM\WebSphere MQ\Qmgrs\TC!QMGR\@ipcc
```

Websphere MQ 8.0:

```
C:\ProgramData\IBM\MQ\qmgrs\TC!QMGR\@ipcc
```

On the TC/LINK-MQ computer, the destination directory is defined by the environment variable MQCHLLIB. On Websphere MQ 7.5 and before the default is:

```
C:\Program Files (x86)\IBM\WebSphere MQ
```

Websphere MQ 8.0:

```
C:\ProgramData\IBM\MQ
```

Note: After copying the new AMQCLCHL.TAB file and restarting TC/LINK-MQ, the link will still fail. In the MQ client log files (C:\Program Files (x86)\IBM\WebSphere MQ\errors or C:\ProgramData\IBM\MQ\WebSphere MQ\errors on version 8.0 or later), you should find a related “CipherSpec” error message.

8.10.3 Creating an SSL Server Certificate on the MQ Server

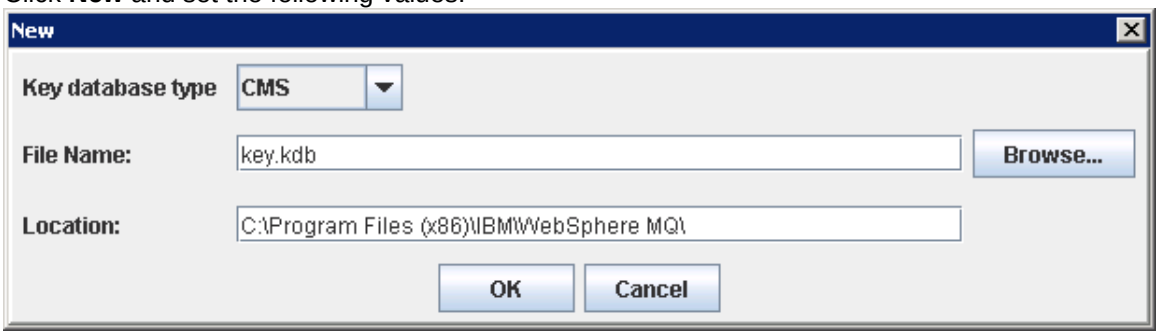
8.10.3.1 Creating a Key Database File

You need to create a queue manager certificate store for the queue manager. Use the IBM Key Management tool to create a certificate store. The tool can be found here:

```
C:\Program Files (x86)\IBM\WebSphere MQ\bin\strmqikm.exe
```

Start the IBM Key Management tool in elevated mode (Run as administrator).

Click **New** and set the following values:



- Key database type: CMS
- File name: key.kdb
- Location: C:\Program Files (x86)\IBM\WebSphere MQ\

Click **OK** to continue.

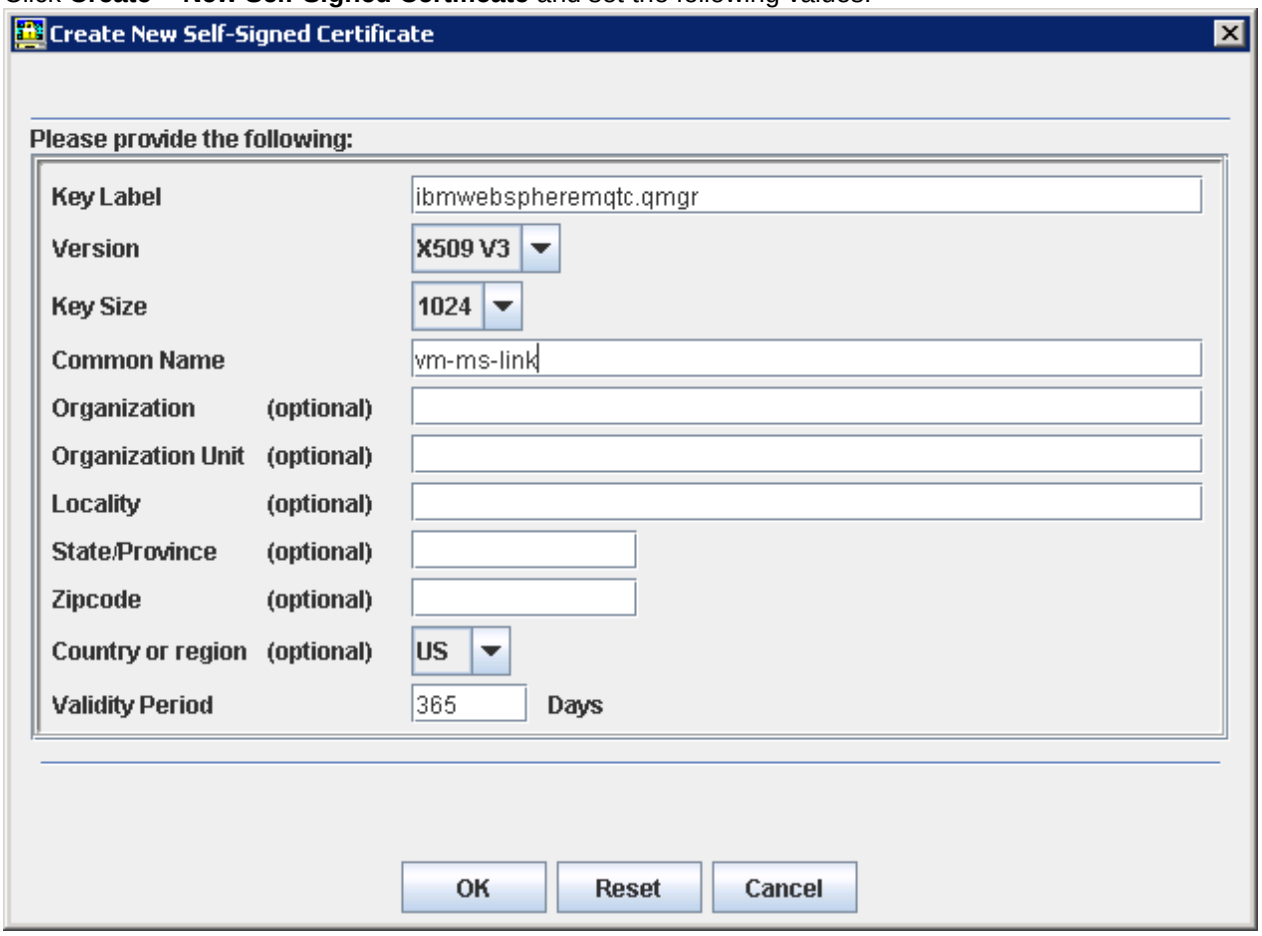
Enter a password and select **Stash the password to a file**.

Click **OK** to continue.

8.10.3.2 Creating a Queue Manager Self-Signed Certificate

The following Key Label value must be in the form <ibmwebspheremq<qmname lowercase>>, e.g.: ibmwebspheremqtc.qmgr

Click **Create**=>**New Self-Signed Certificate** and set the following values:



- Key label: ibmwebspheremqtc.qmgr
- Common Name: <fully qualified domain name or computer name> (choose an appropriate value)

Click **OK** to continue.

You now should have a personal certificate listed as * ibmwebspheremqtc.qmgr. Click **Extract certificate** to extract the self-signed certificate to a file. Set the following values:

- Data type: Binary DER data
- Certificate file name: TcQmgr.der

Click **OK** to continue.

8.10.4 Creating an SSL Server Certificate on the TC/LINK-MQ Computer

Copy the extracted certificate to the TC/LINK-MQ machine. As on the MQ Server, use the IBM Key Management tool to create a Key Database file in the root folder of your MQ Client installation:

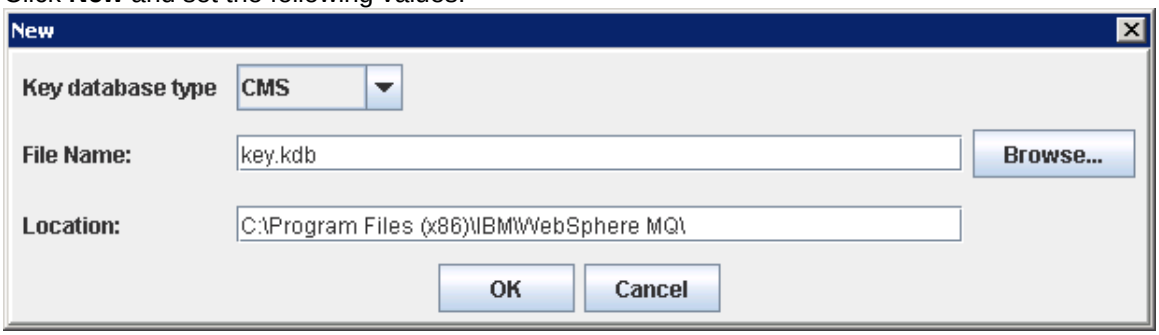
```
C:\Program Files (x86)\IBM\WebSphere MQ
```

8.10.4.1 Creating a Key Database File

Start the IBM Key Management tool in elevated mode (Run as administrator). The tool can be found here:

```
C:\Program Files (x86)\IBM\WebSphere MQ\bin\strmqikm.exe
```

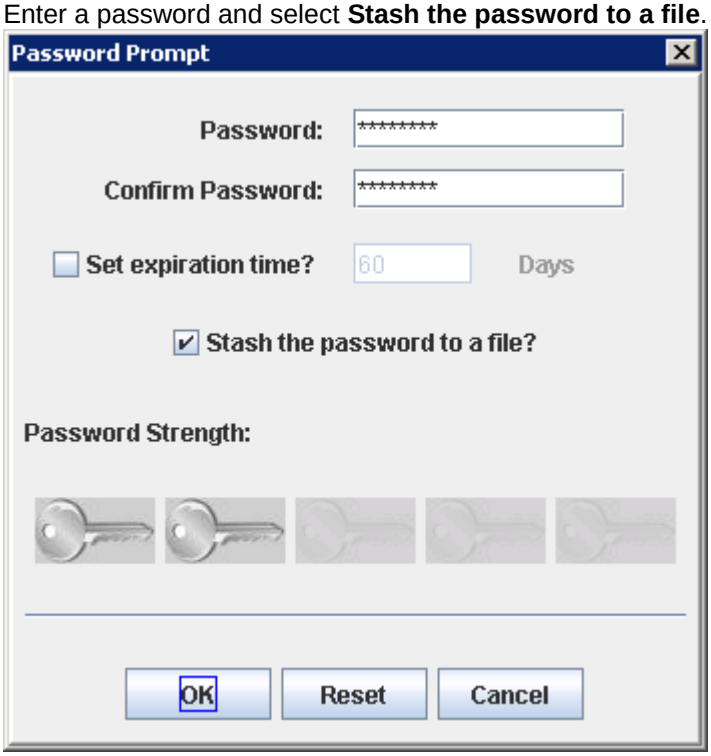
Click **New** and set the following values:



- Key database type: CMS
- File name: key.kdb
- Location: C:\Program Files (x86)\IBM\WebSphere MQ\

Click **OK** to continue.

Enter a password and select **Stash the password to a file**.



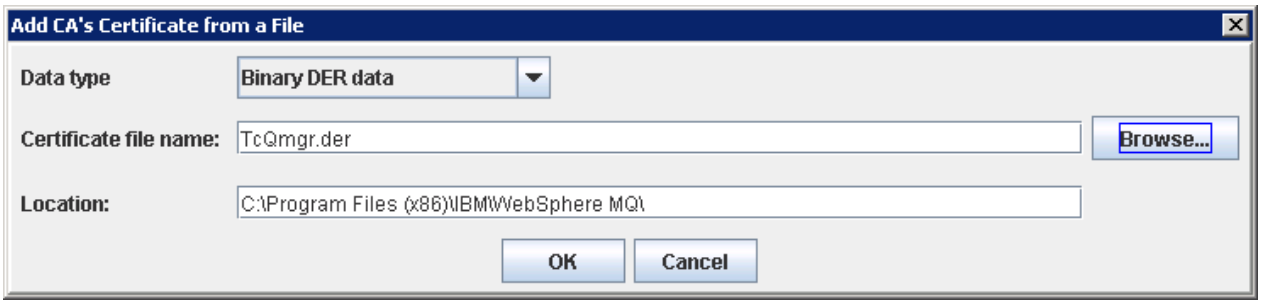
Click **OK** to continue.

8.10.4.2 Add the Certificate to the TC/LINK-MQ Key Database File

Select the drop-down box under the label Key database content.

Select **Signer Certificates**.

Click **Add**. You will be prompted for the location of the certificate you wish to add. This certificate will either be the queue managers certificate if you are using self-signed certificates for testing, or the certificate of the CA that issued your queue managers certificate. This example uses the self signed certificate:



Click **OK**. You will be prompted for a label. Enter ibmwebspheremqtc.qmgr.

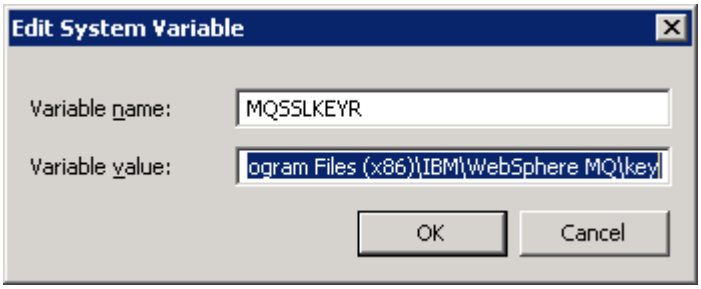
Click **OK** to add the certificate.

8.10.4.3 Create the Environment Variable MQSSLKEYR

The MQ Client checks the environment variable MQSSLKEYR to find the Key Database file. Create a new System Environment variable and set the path to the Key Database file without file extension:

MQSSLKEYR

```
C:\Program Files (x86)\IBM\WebSphere MQ\key
```



Restart TCSRV (not only TC/LINK-MQ). The process should now start successfully.

8.10.5 Troubleshooting

Verbose error descriptions can be found in the IBM WebSphere log files on the client and on the server:

TC/LINK-MQ computer:

```
C:\Program Files (x86)\IBM\WebSphere MQ\errors
```

MQ Server:

```
C:\Program Files (x86)\IBM\WebSphere MQ\Qmgrs\TC!QMGR\errors
```

8.10.5.1 Revocation Status Check Failed

MQ version 7.0.1 reports a problem in checking the revocation list of the certificates. An IBM support article offers more information about the subject:

http://www-01.ibm.com/support/docview.wss?uid=swg21399255

As a temporary workaround, you can add the following two lines to the MQ client ini file (C:\Program Files (x86)\IBM\WebSphere MQ\mqclient.ini):

```
SSL:
OCSPAuthentication=OPTIONAL
```

8.10.5.2 MQ Connection Lost with MQ Error AMQ9211 After Document Conversion Timeout

In MQ 7.5, an SSL session gets closed when there is no activity between the sender and receiver. An IBM support article offers more information about the subject:

http://www-01.ibm.com/support/docview.wss?uid=swg1IC88852

Installing the 7.5.0.1 fix pack at the client computer fixed the problem