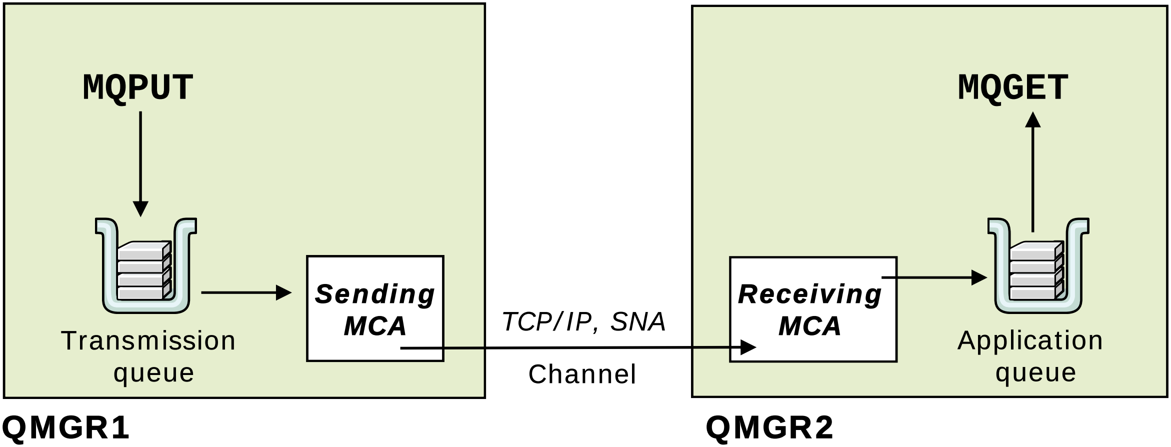
Implementing distributed queuing

Unit objectives

After completing this unit, you should be able to:

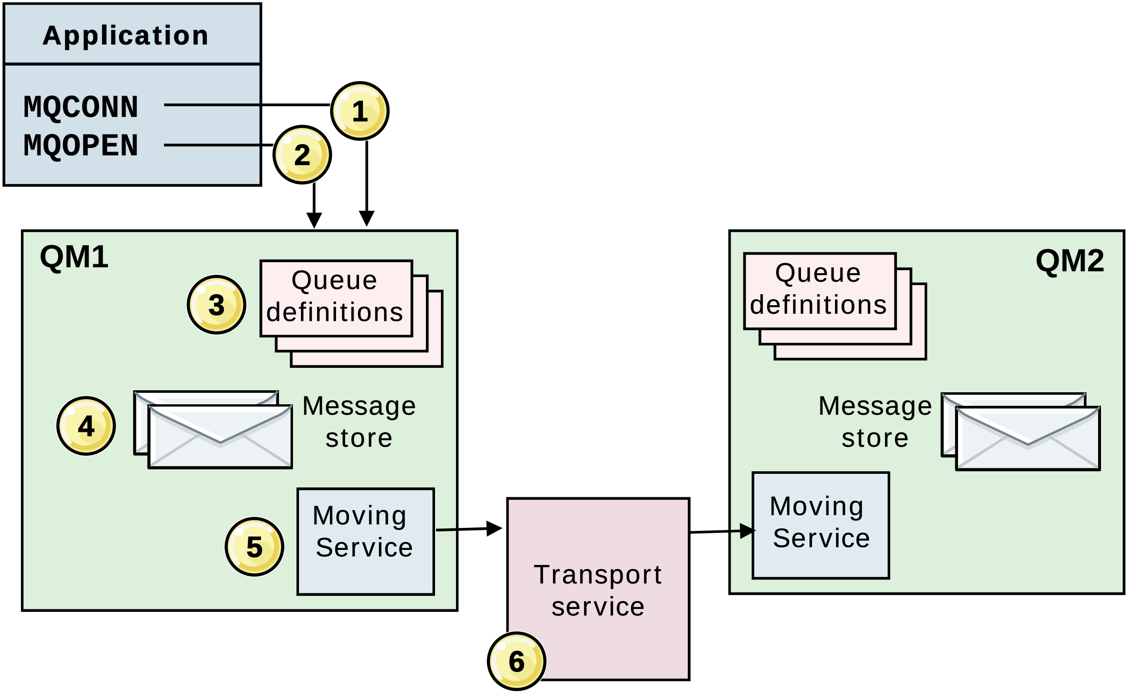
- Diagram the connection between two queue managers by using the required components
- Configure IBM MQ channels
- Start and stop channels
- Identify channel states
- Access remote queues
- List considerations for data conversion
- Use the dead-letter queue to find messages that could not be delivered

Message channel



- One-way link that connects two queue managers by using a *message channel agent* (MCA)

Distributed component overview



Application

MQCONN

MQOPEN

1

2

QM1

3

Queue definitions

4

Message store

5

Moving Service

Transport service

6

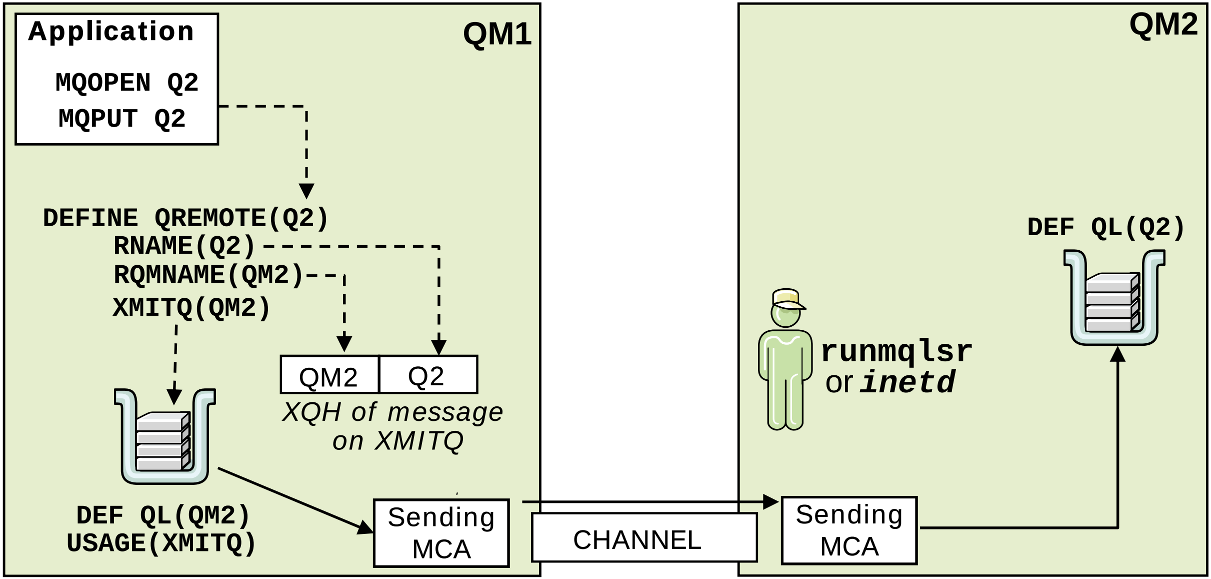
QM2

Queue definitions

Message store

Moving Service

Distributed queuing overview



QM1

Application

```
MQOPEN Q2
MQPUT Q2
```

```
DEFINE QREMOTE(Q2)
     RNAME(Q2)
     RQMNAME(QM2)
     XMITQ(QM2)
```

QM2	Q2

XQH of message on XMITQ

```
DEF QL(QM2)
USAGE(XMITQ)
```

Sending MCA

CHANNEL

QM2

```
DEF QL(Q2)
```

runmqlsr or ***inetd***

Sending MCA

```
DEFINE CHANNEL(QM1.QM2)
       CHLTYPE(SDR)
       CONNAME(QM2_address)
       XMITQ(QM2)
       TRPTYPE(TCP)

START CHANNEL (QM1.QM2)
```

```
DEFINE CHANNEL (QM1.QM2)
       CHLTYPE (RCVR)
       TRPTYPE(TCP)
```

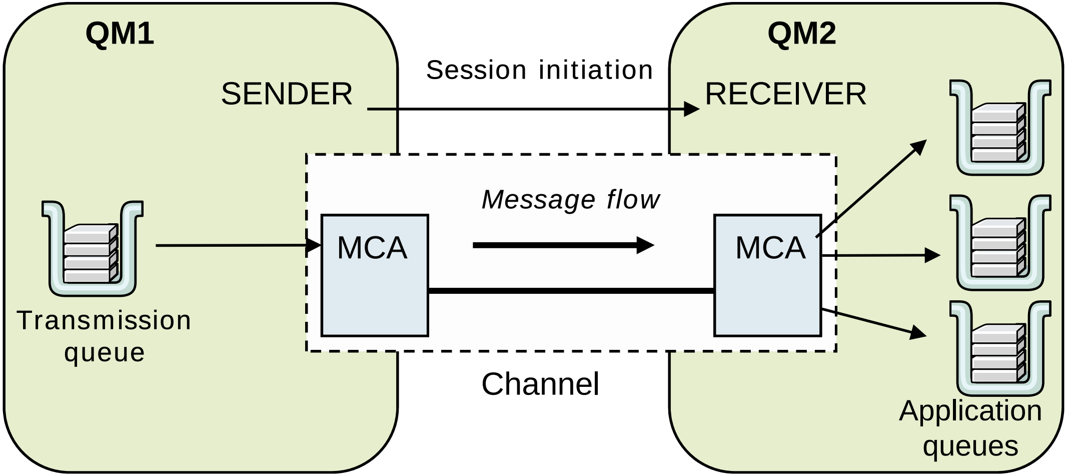
Distributed queuing components

- Queue that is identified by:
  - Name of the queue
  - Name of the queue manager that owns the queue
- Message channels carry messages from one queue manager to another
- Message channel agents (MCAs)
  - Control the sending and receiving of messages
  - One at each end of the channel
- Transmission queues store messages for a remote queue manager
- Channel initiators and listeners act as trigger monitors for the sender channels
- Channel-exit programs

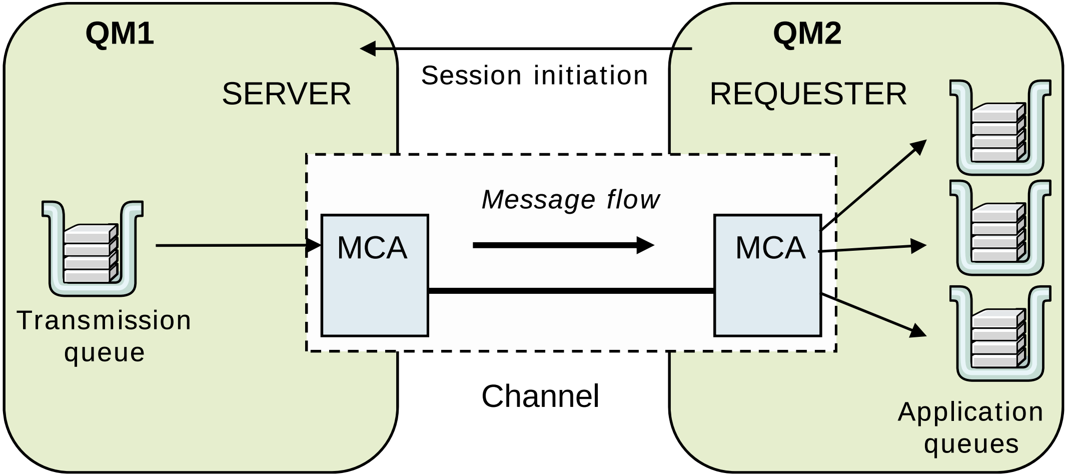
Basic channel types

- For distributed queuing:
  - Sender
  - Server
  - Receiver
  - Requester
- For clustered environments
  - Cluster-sender
  - Cluster-receiver
- For IBM MQ clients
  - Client-connection
  - Server-connection

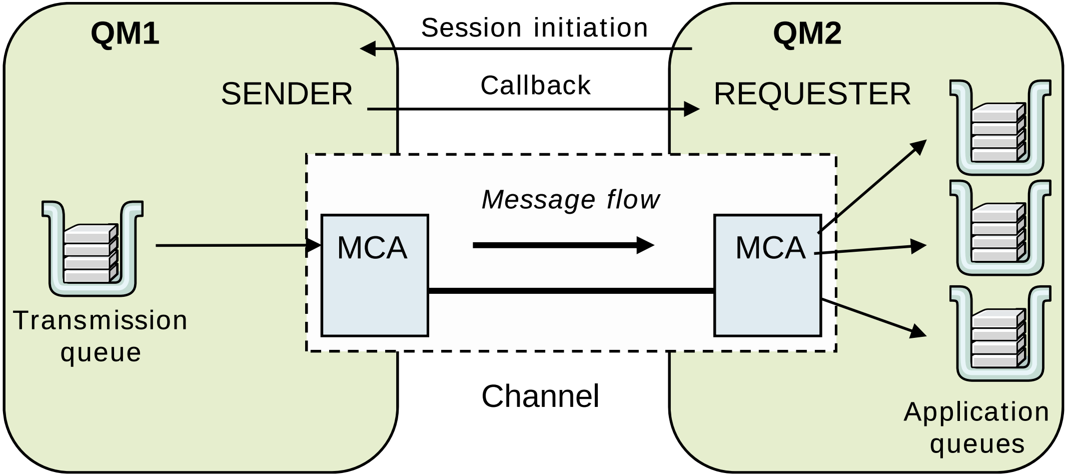
Sender-receiver channels



Requester-server channels



Requester-sender channels



Choosing a transmission queue

1. Specify the transmission queue in local definition of a remote queue.

```
DEFINE QREMOTE(BBB) RNAME(YYY) +
RQMNAME(QM2) XMITQ(EXPRESS)
```

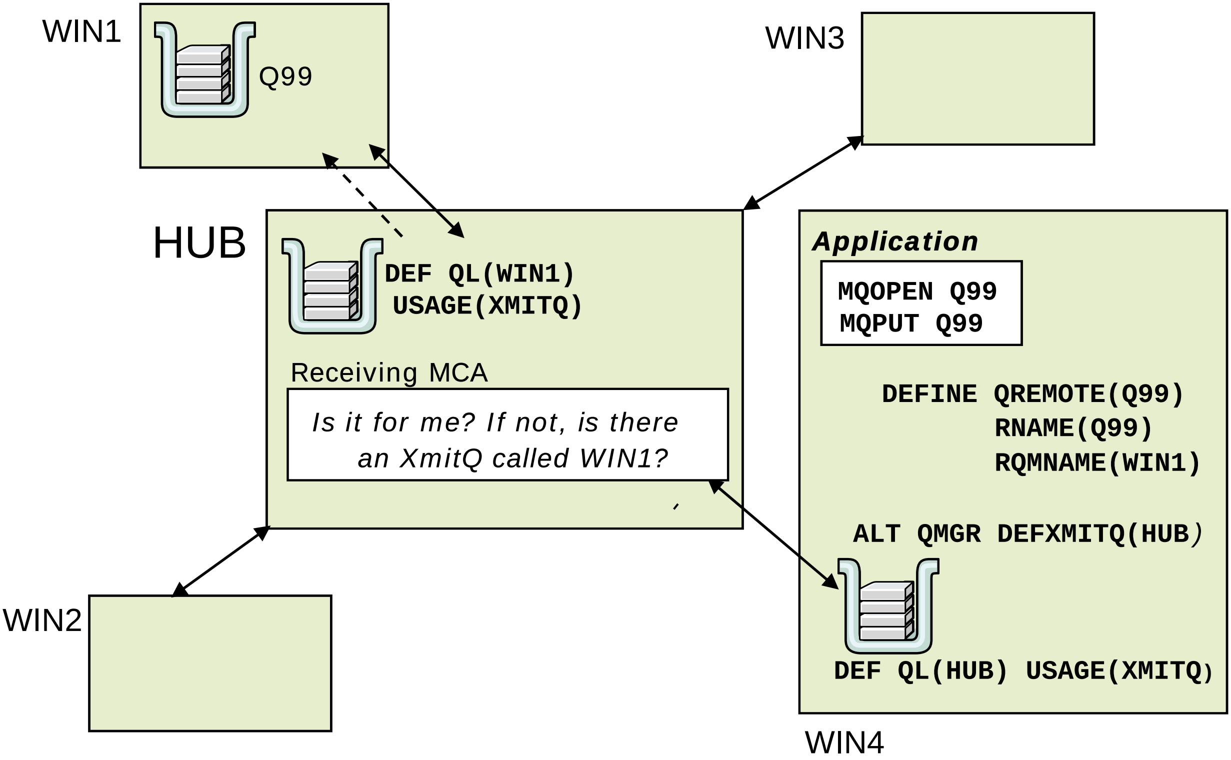
2. The name of the transmission queue is inferred from the name of the remote queue manager.

3. Specify the default transmission queue attribute on the queue manager.

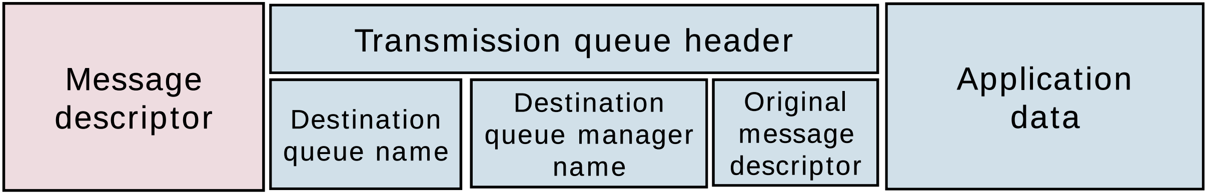
```
ALTER QMGR DEFXMITQ(HOST)
```

4. Error (the MQOPEN fails).

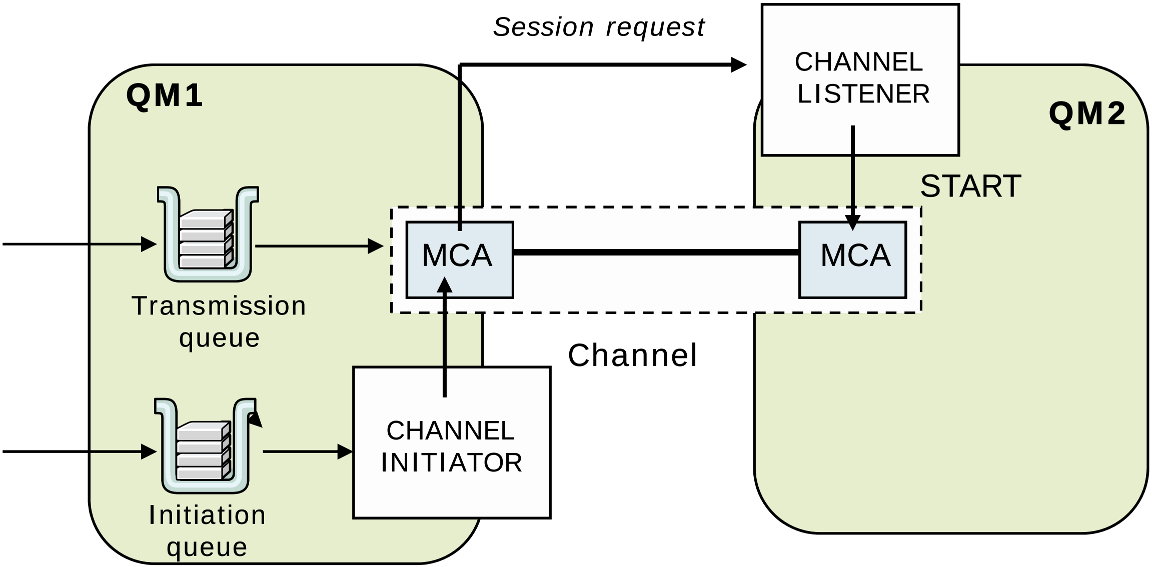
Example of using a default transmission queue



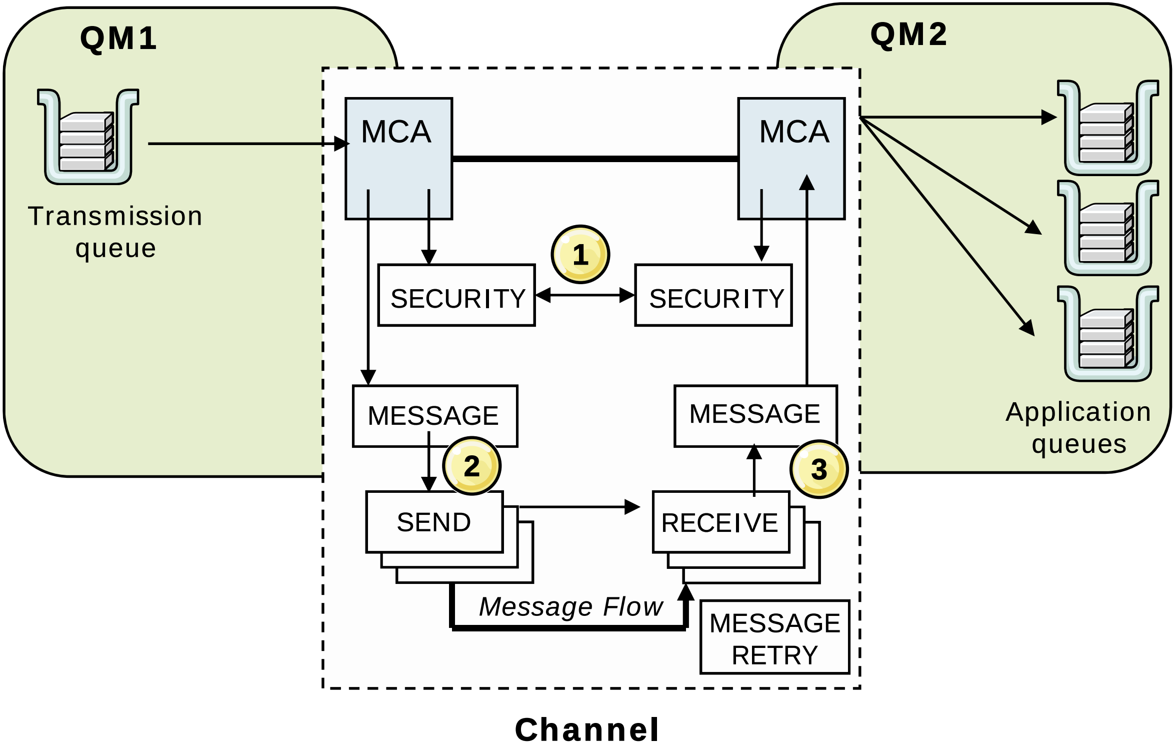
Transmission queue header


Message descriptor
Transmission queue header
Destination queue name
Destination queue manager name
Original message descriptor
Application data

Channel initiators and listeners



Channel-exit programs



Configuration file stanzas for distributed queuing

- Stanzas in the queue manager configuration file (**`qm.ini)`** for distributed queuing
  - CHANNELS
  - TCP
  - LU62
  - NETBIOS
  - SPX (Windows XP Professional and Windows 2003 Server only)
  - EXITPATH

- Edit configuration files either:
  - Automatically, by using commands or IBM MQ Explorer options that change the configuration of queue managers
  - Manually, by using a text editor

Example queue manager configuration file stanzas

```
CHANNELS:
MaxChannels=n        ; Maximum channels allowed. Default is 100
MaxActiveChannels=n  ; Maximum active channels allowed at any time.
                     ; Default is the value of MaxChannels
MaxInitiators=n      ; Maximum initiators allowed. Default and maximum
                     ; value is 3
MQIBINDTYPE=type1    ; Whether the binding for applications is “fastpath”
                     ; or “standard”. Default is “standard”.
ThreadedListener=    ; “YES” to run all channels run as threads of a single
                     ; job. “NO” to start a new responder job for each
                     ; inbound TCP/IP channel. Default is “NO”.
AdoptNewMca=chltype  ; Stops previous process if channel fails to start.
                     ; The default is “NO”.
AdoptNewMcaTimeout=  ; Amount of time that the new process waits for the
                     ; old process to end. Default is 60.
AdoptNewMcaCheck=    ; Type of checking required when enabling AdoptNewMca.
          typecheck  ; Default is “NAME”,“ADDRESS”, and “QM”.

TCP:                 ; TCP entries
Port=n               ; Port number, the default is 1414
KeepAlive=NO         ; Switch TCP/IP KeepAlive “ON” or “OFF”
ListenerBacklog=n    ; Maximum outstanding connection requests.
ConnectTimeout=n     ; Seconds before an attempt to connect socket times
                     ; out. Default of zero specifies that there is no
                     ; connect timeout.
```

Minimum channel definitions for remote communication

- On the source queue manager, channel type of SENDER
  - XMITQ specifies the name of the transmission queue
  - CONNAME defines the connection name of the partner system
  - TRPTYPE specifies the transport protocol (default is ‘TCP’)

- On the target queue manager, channel type RECEIVER
  - Same name as sender channel
  - TRPTYPE to specify the transport protocol (default is ‘TCP’)

- Use the **`ping`** command to test the channel

Define channel command

```
DEF CHL('channel') CHLTYPE(string) CONNAME(string) +
  TRPTYPE(string) XMITQ(string)
```

- Required for definition

(*channel*) Channel name up to 20 characters. Must be the first parameter.

CHLTYPE(*string*) Channel type

Sender:	**SDR**
Receiver:	**RCVR**
Server:	**SVR**
Requester:	**RQSTR**

CONNAME(*string*) Communication connection identifier for **SDR** and **RQSTR**, optional for **SVR**

TRPTYPE Transport type: **LU62, NETBIOS, SPX, TCP** (default)

XMITQ(*string*) Name of the transmission queue from which messages are retrieved for **SDR** and **SVR**

Defining channels example

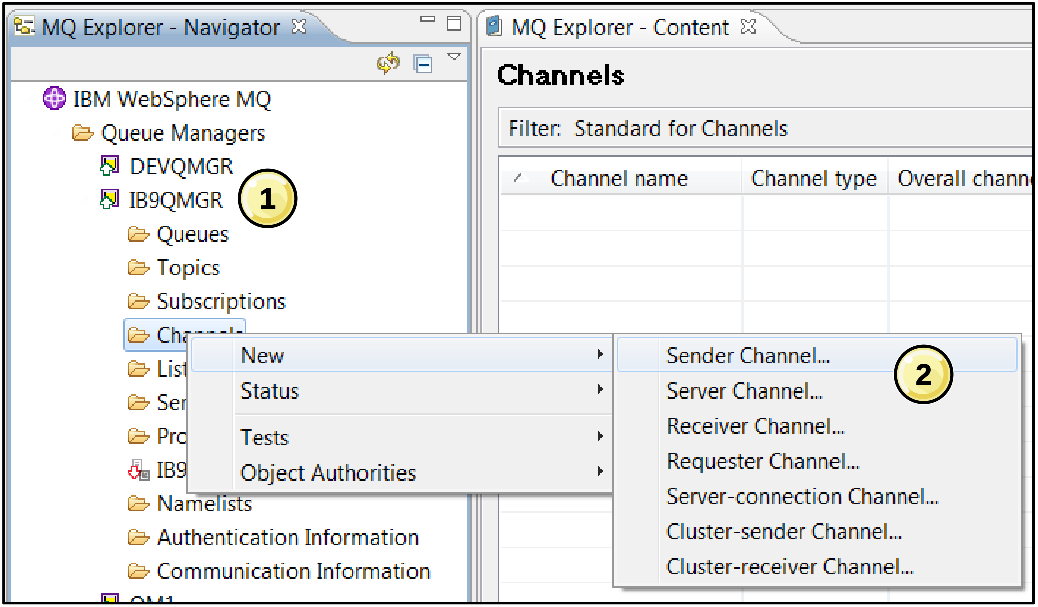
Hursley

```
DEF CHL('Hursley_Dallas') +
CHLTYPE(SDR) TRPTYPE(TCP) +
CONNAME(‘9.84.100.1(1414)’) +
XMITQ('Dallas')

 DEF QL('Dallas') USAGE(XMITQ)


   -------------------------


 DEF CHL('Dallas_Hursley') +
 CHLTYPE(RCVR) TRPTYPE(TCP)
```

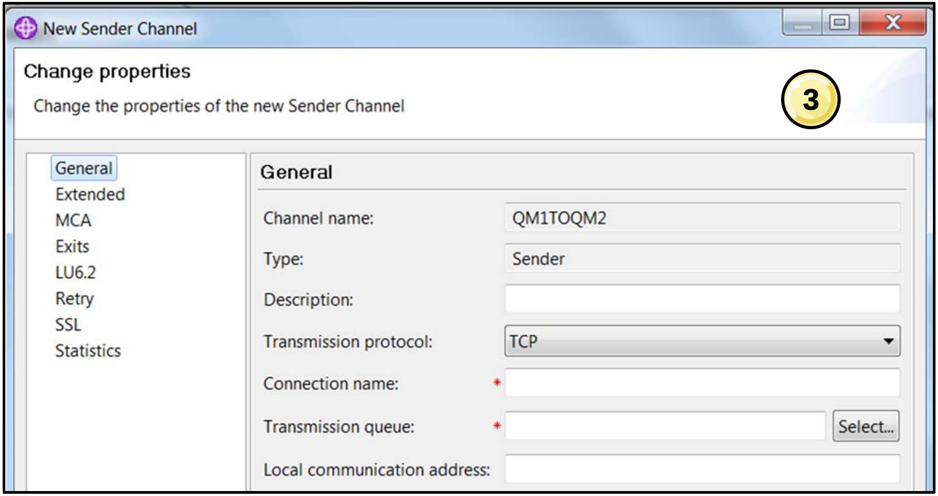
Dallas

```
DEF CHL('Hursley_Dallas') +
CHLTYPE(RCVR) TRPTYPE(TCP)


   -------------------------


DEF CHL('Dallas_Hursley') +
CHLTYPE(SDR) TRPTYPE(TCP) +
CONNAME(‘9.20.31.5(1414)’) +
XMITQ('Hursley')

DEF QL('Hursley') USAGE(XMITQ)
```

Defining channels in IBM MQ Explorer (1 of 2)



Defining channels in IBM MQ Explorer (2 of 2)



New Sender Channel

Change properties

Change the properties of the new Sender Channel

General

Channel name: QM1TOQM2

Type: Sender

Description:

Transmission protocol: TCP

Connection name: *

Transmission queue: *

Local communication address:

- **MCA**: Specify how the MCA program of the channel definition runs
- **Exits**: Configure any user exits
- **LU6.2**: Configure the MCA if the MCA uses the LU6.2 transport protocol
- **Retry**: Configure the channel behavior if the channel cannot connect to the remote queue manager
- **SSL**: Specify the SSL settings for this end of the channel
- **Statistics**: Control the collection of statistics and monitoring data for the channel

Controlling the listener process

- Start the IBM MQ listener: **`runmqlsr`**
  - Run synchronously and waits until the listener process has finished before returning to the caller
  - Must identify the transmission protocol as TCP/IP, SNA LU 6.2 (Windows only), NetBIOS (Windows only), or SPX (Windows only)

- End the IBM MQ listener: **`endmqlsr`**

Listener object

- Create a listener object:
  - Optionally, specify **CONTROL(QMR)** so that the listener starts and stops when the queue manager starts and stops

```
DEFINE LISTENER(LISTENER.TCP) TRPTYPE(TCP) +
     PORT(1414) +
     CONTROL(QMGR) +
     REPLACE
```

- Start the listener:

```
START LISTENER(LISTENER.TCP)
```

Starting a message channel

- IBM MQ commands:

```
PING CHANNEL(QMA_QMB)
START CHANNEL(QMA_QMB)
```

- Start a sender or requester channel
  - From a command: **runmqchl -c *Channel* –m *Qmgr***
  - From IBM MQ Explorer

- Channel attributes that are evaluated when channel is started:

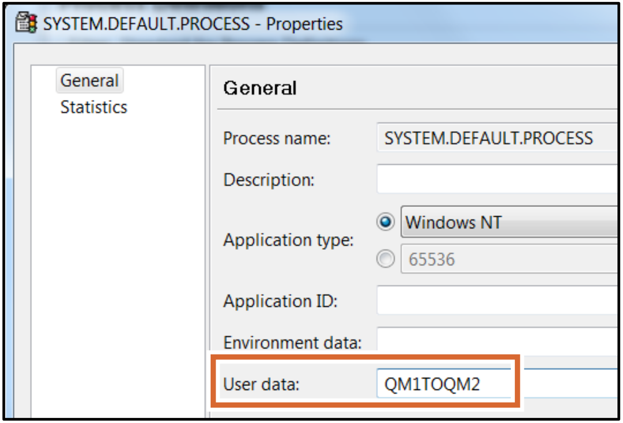
BATCHSZ	Batch size
MAXMSGL	Maximum message length
SEQWRAP	Sequence number wrap
HBINT	Heartbeat interval
NPMSPEED	Non-persistent message speed

Reasons why a channel might fail to start

- Definitions at both ends of a channel do not specify the same channel name
- Channel is not defined at both ends with compatible types
- A sequence number mismatch
  - Deleted a channel definition at one end of a message channel and then redefined it
  - Deleted and created a queue manager again

Channel initiator

- Two main functions:
  - Triggering a channel
  - Try channels again
- Options to starts an MCA at sending end of a message channel:
  - Specify the channel name in **User data** attribute of the process object
  - Specify the channel name in **Trigger data** attribute of the transmission queue



SYSTEM.DEFAULT.PROCESS - Properties

General

Process name: SYSTEM.DEFAULT.PROCESS

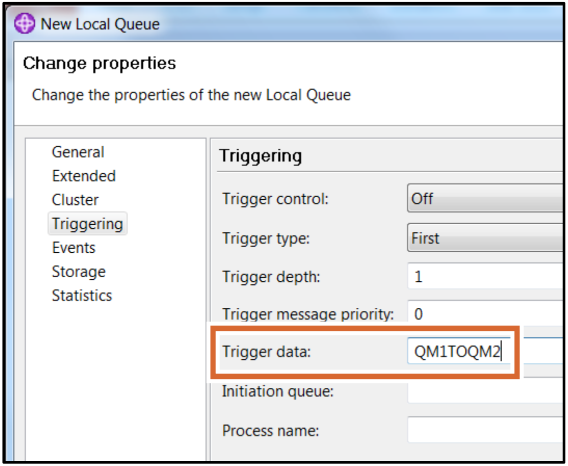
Description:

Application type: ● Windows NT ○ 65536

Application ID:

Environment data:

User data: QM1TOQM2



New Local Queue

Change properties

Change the properties of the new Local Queue

Triggering

Trigger control: Off

Trigger type: First

Trigger depth: 1

Trigger message priority: 0

Trigger data: QM1TOQM2

Initiation queue:

Process name:

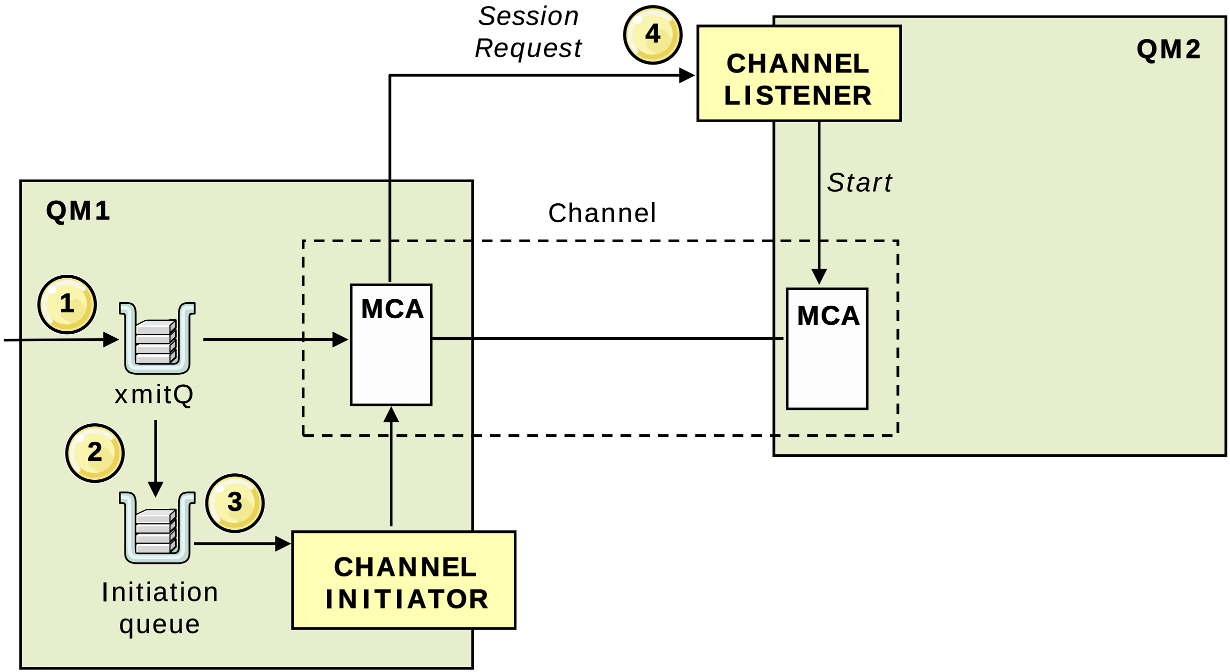
Channel control parameters

- Specified on the **DEFINE CHANNEL** or **ALTER CHANNEL** command

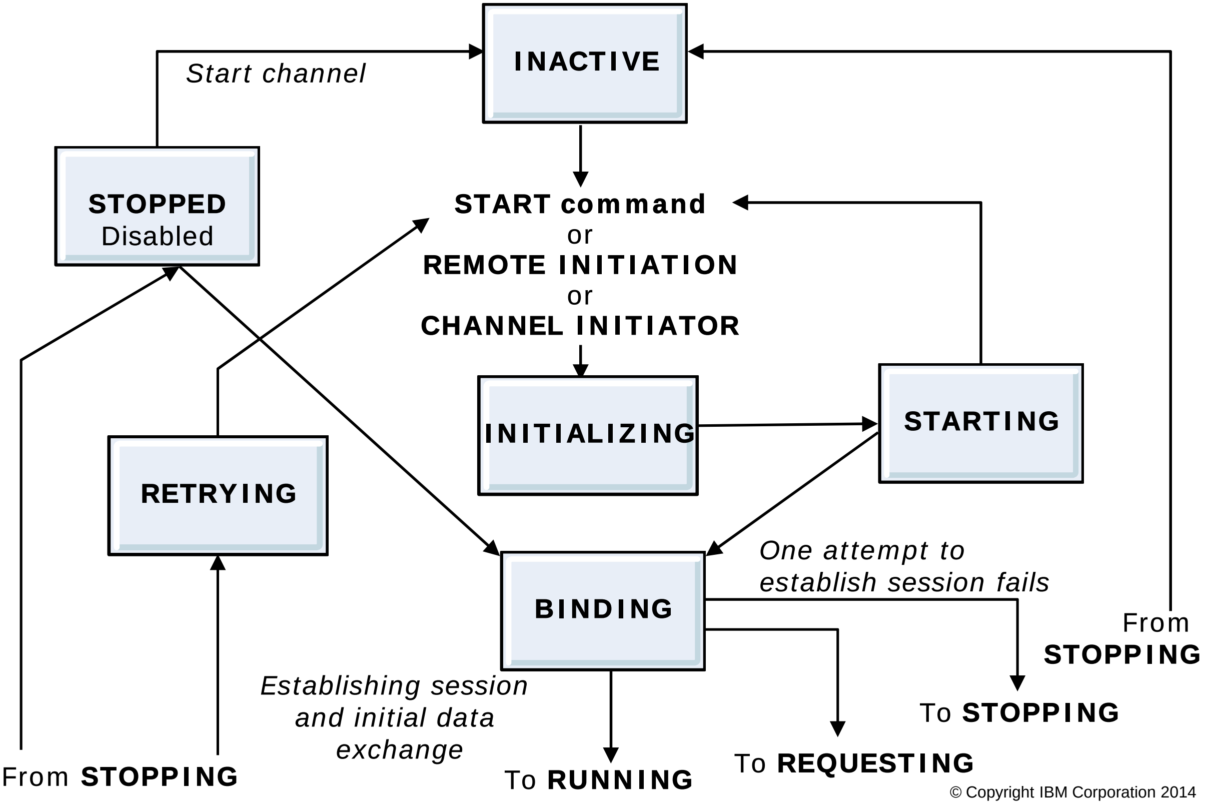
DISCINT	Disconnect interval
SHORTRTY, SHORTTMR	Short retry count, short retry interval
LONGRTY, LONGTMR	Long retry count, long retry interval
MRRTY, MRTMR	Message retry count, message retry interval
MCATYPE	Message channel agent type of **PROCESS** or **THREAD**
BATCHINT	Batch interval

- Can be configured in IBM MQ Explorer **Channel > Properties**

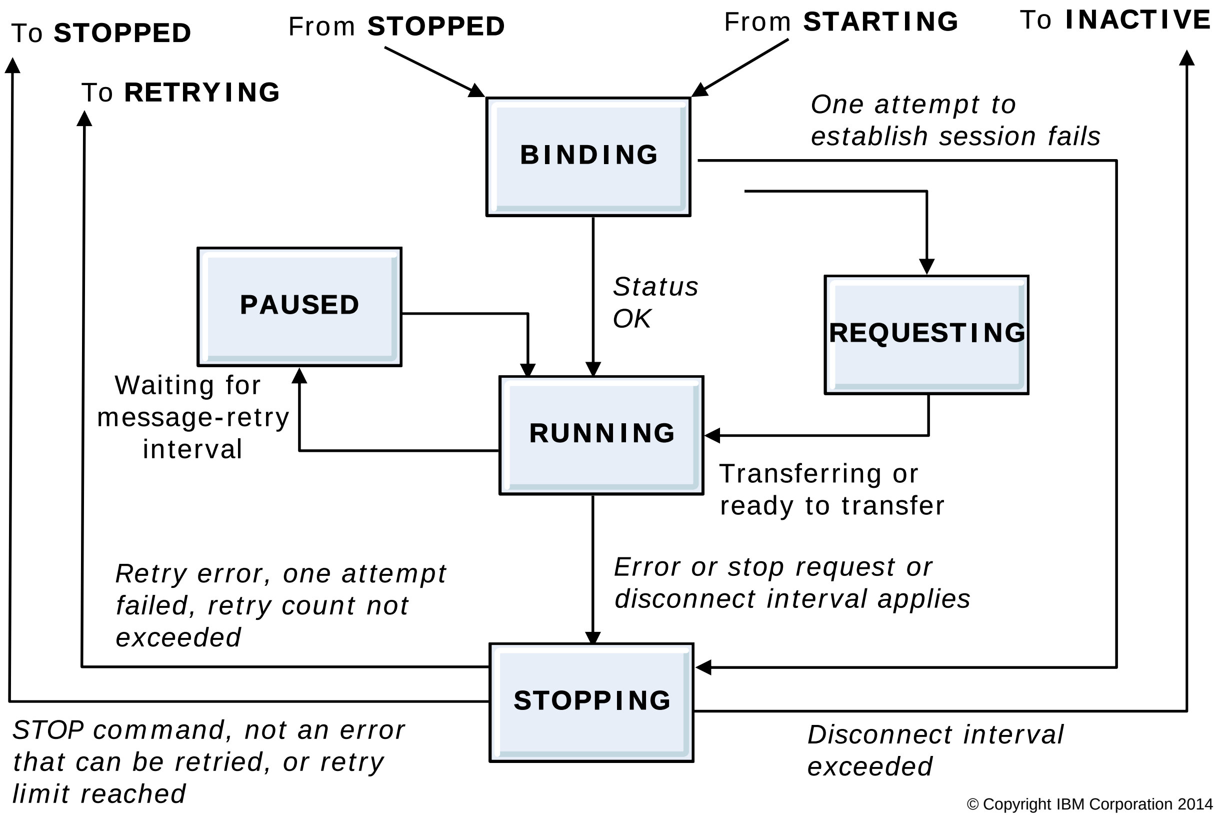
Channel initiator triggering



Channel states: DISPLAY CHSTATUS command (1 of 2)



Channel states: DISPLAY CHSTATUS command (2 of 2)


To **STOPPED**
From **STOPPED**
From **STARTING**
To **INACTIVE**
To **RETRYING**
BINDING
One attempt to establish session fails
PAUSED
Status OK
REQUESTING
Waiting for message-retry interval
RUNNING
Transferring or ready to transfer
Error or stop request or disconnect interval applies
Retry error, one attempt failed, retry count not exceeded
STOPPING
STOP command, not an error that can be retried, or retry limit reached
Disconnect interval exceeded

Queue definitions for distributed queuing

- Local definition of a remote queue

  Example: **`DEFINE QREMOTE(BBB) RNAME(YYY) RQMNAME(QM2)`**

- Transmission queue must be created at the sending end of each message channel
  - **`USAGE(XMITQ)`** indicates its purpose
  - It can have any of the attributes of a local queue

  Example: **`DEFINE QLOCAL(QM2) USAGE(XMITQ)`**

Define remote queue (QREMOTE)

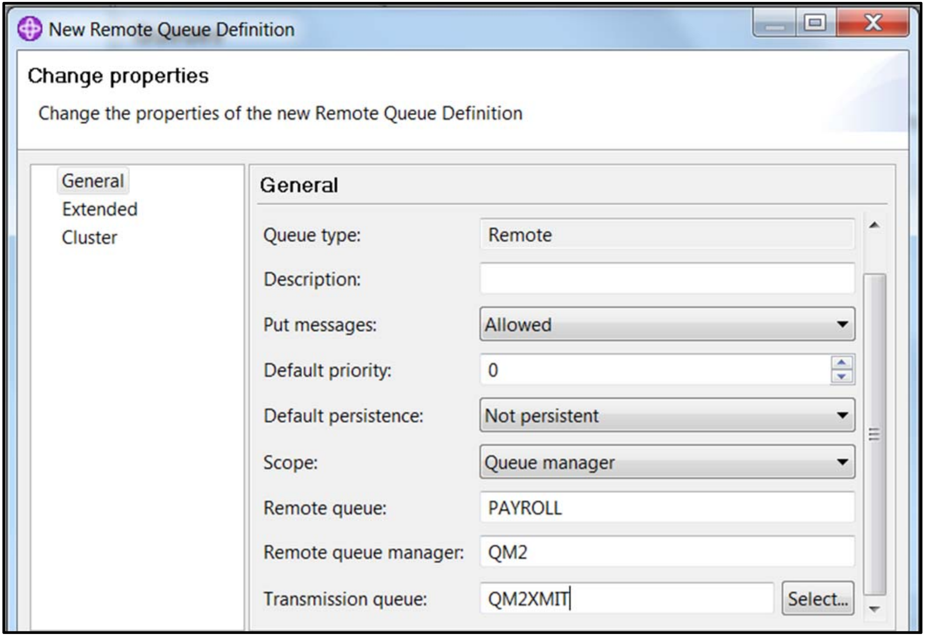
- Another queue manager owns remote queues
- Define a remote queue definition

Example:

Define a remote queue that is named PAYROLL.QUERY on the local queue manager. The physical queue is named PAYROLL on a remote queue manager QM2. The local queue manager uses the transmission queue that is named QM2.

```
DEFINE QREMOTE(PAYROLL.QUERY) +
DESCR('Remote queue for QM2') +
REPLACE RNAME(PAYROLL) RQMNAME(QM2) XMITQ(QM2)
```

Using IBM MQ Explorer to define a remote queue



Change properties

Change the properties of the new Remote Queue Definition

General

Queue type: Remote

Description:

Put messages: Allowed

Default priority: 0

Default persistence: Not persistent

Scope: Queue manager

Remote queue: PAYROLL

Remote queue manager: QM2

Transmission queue: QM2XMIT

1. Right-click **Queues**, and then click **New > Remote Queue Definition**.
2. Enter a name for the local definition of the remote queue.
3. Enter the remote queue definition properties.

Managing a remote queue manager with IBM MQ Explorer

1. Create a server connection channel.

   Example command:

   ```
   DEF CHL(SYSTEM.ADMIN.SVRCONN) CHLTYPE(SVRCONN)
   MCAUSER(MQADMIN)
   ```

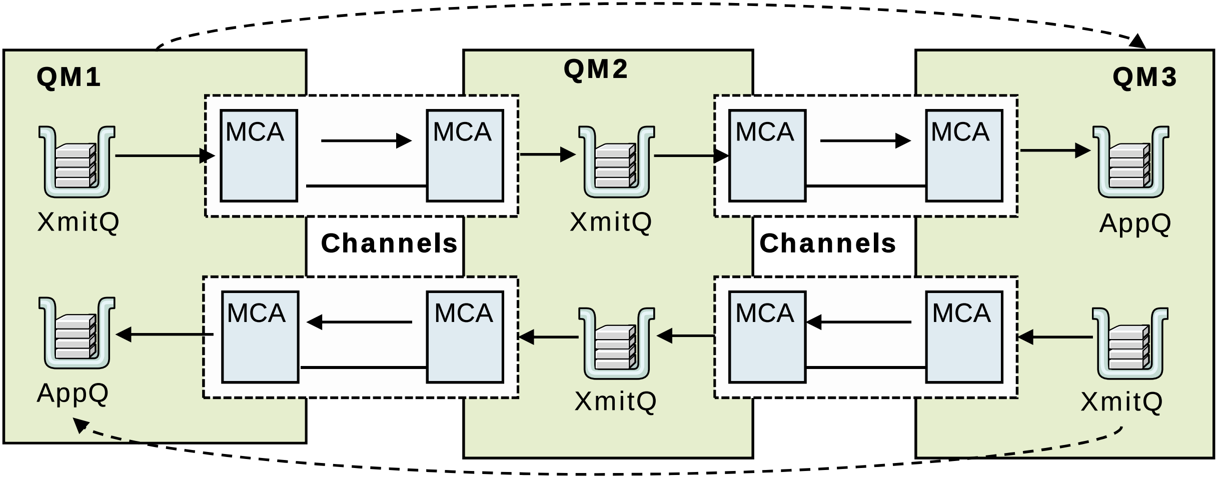
   Where **MQADMIN** is the MQ administrator user ID

2. Create a listener to accept incoming network connections.
3. Start the listener.
4. On local IBM MQ Explorer, right-click **Queue Managers**, click **Add remote queue manager**, and then identify the remote queue manager.

Methods for accessing a remote queue manager

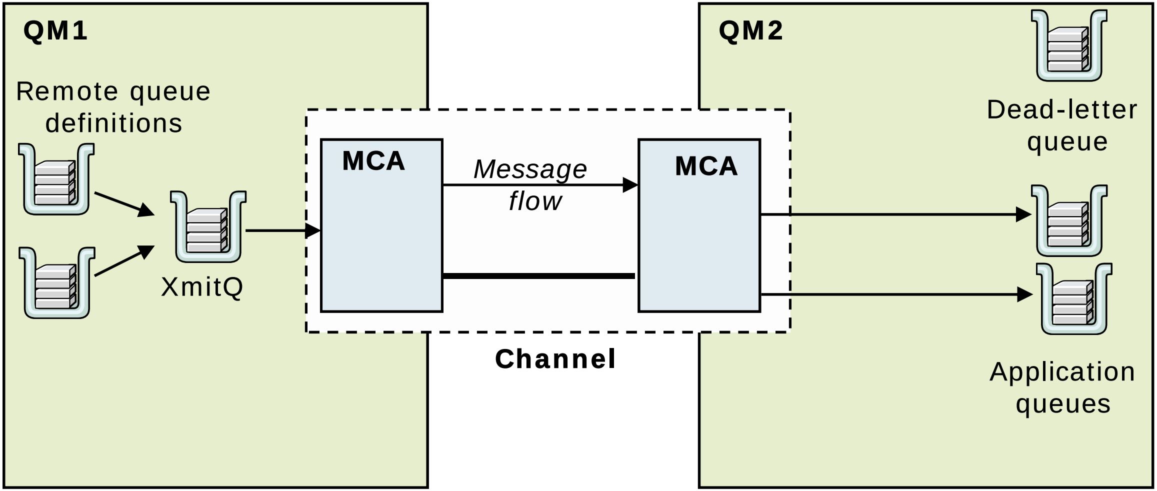
- Multi-hopping
  - Using intermediate queue managers
  - Channel and transmission queue definitions required
- Channel sharing
  - Remote queue definitions specify the transmission queue
  - Using different channels
- Clustering

Multi-hopping



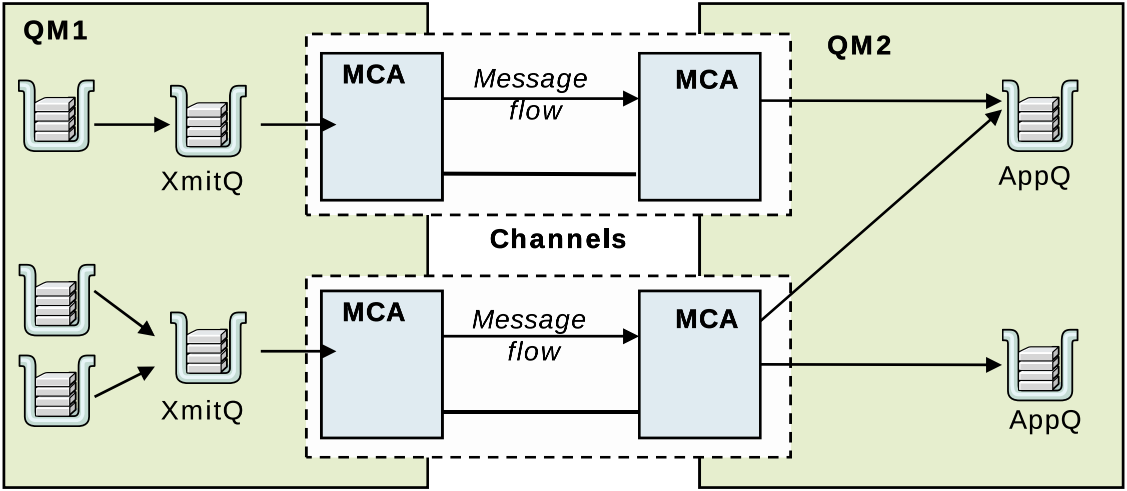
Pass through one or more intermediate queue managers when there is no direct communication link between the source queue manager and the target queue manager.

Channel sharing



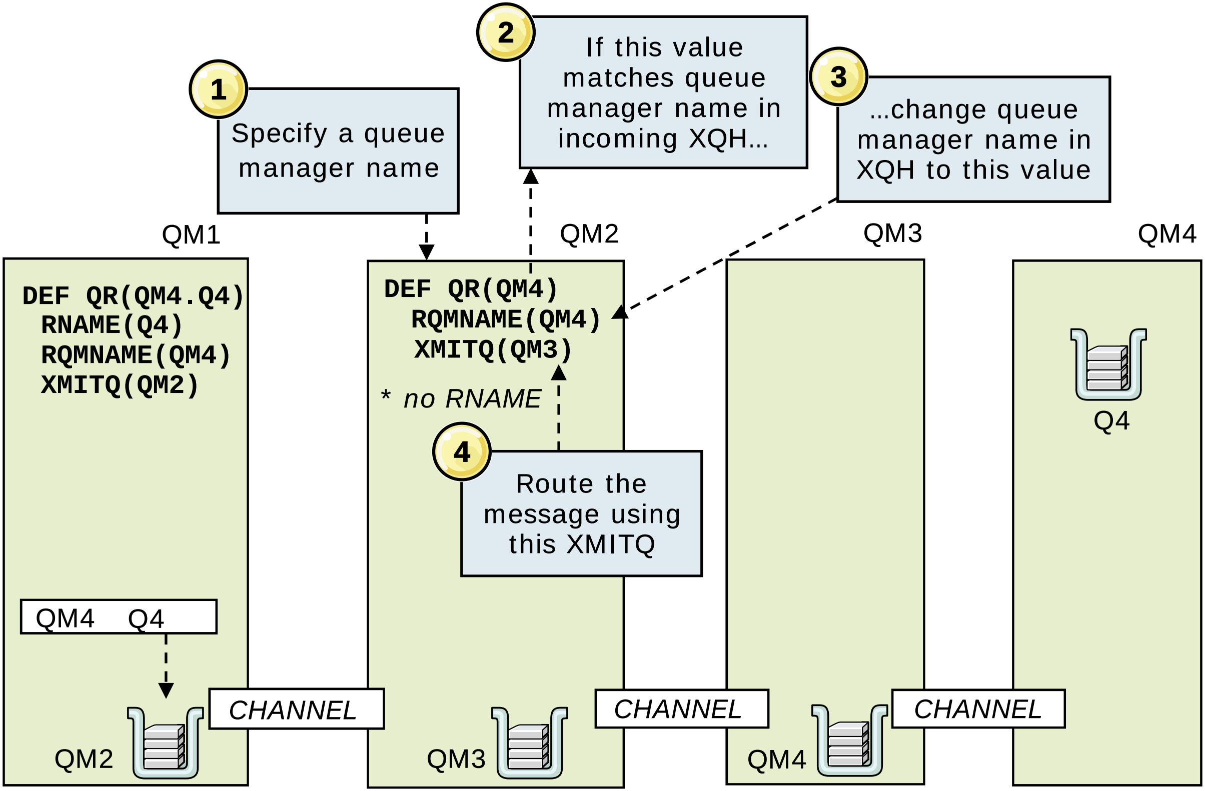
All messages from all applications that address queues at the same remote location send their messages through the same transmission queue.

Using different channels



If you have messages of different types to send between two queue managers, you can define more than one channel between the them.

Using a queue manager alias



1 Specify a queue manager name

2 If this value matches queue manager name in incoming XQH...

3 ...change queue manager name in XQH to this value

4 Route the message using this XMITQ

QM1

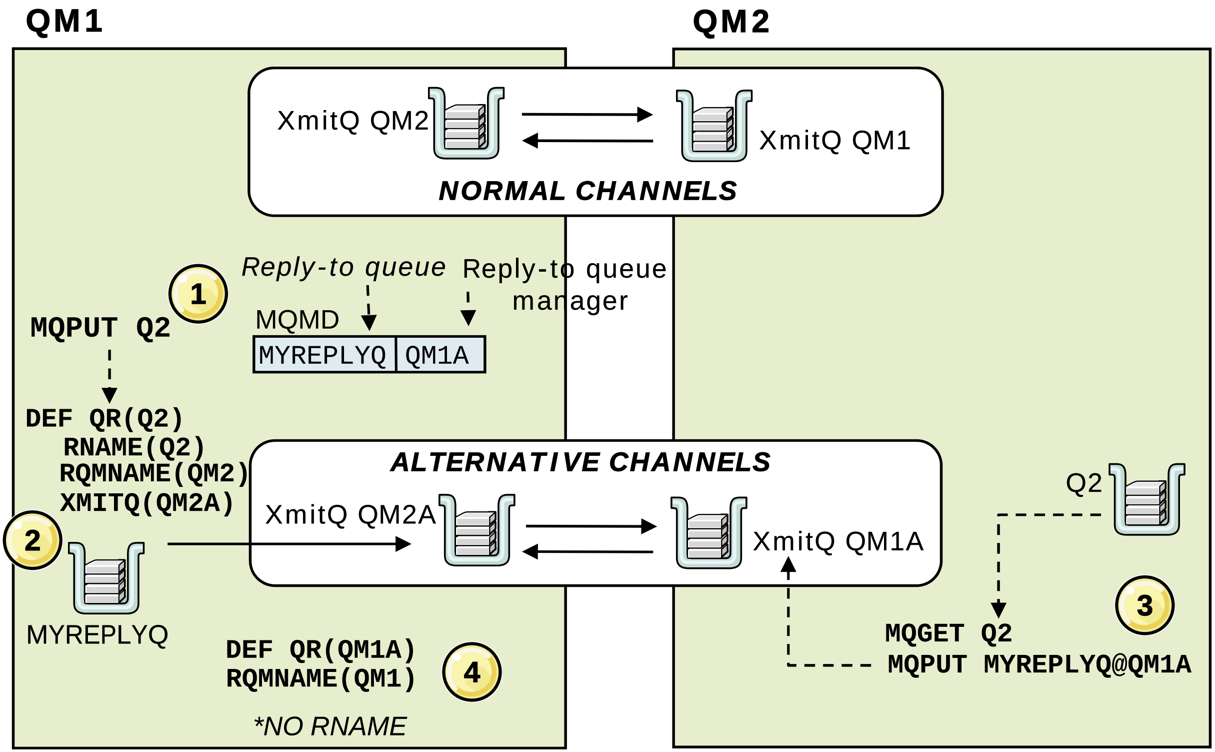
```
DEF QR(QM4.Q4)
  RNAME(Q4)
  RQMNAME(QM4)
  XMITQ(QM2)
```

QM2

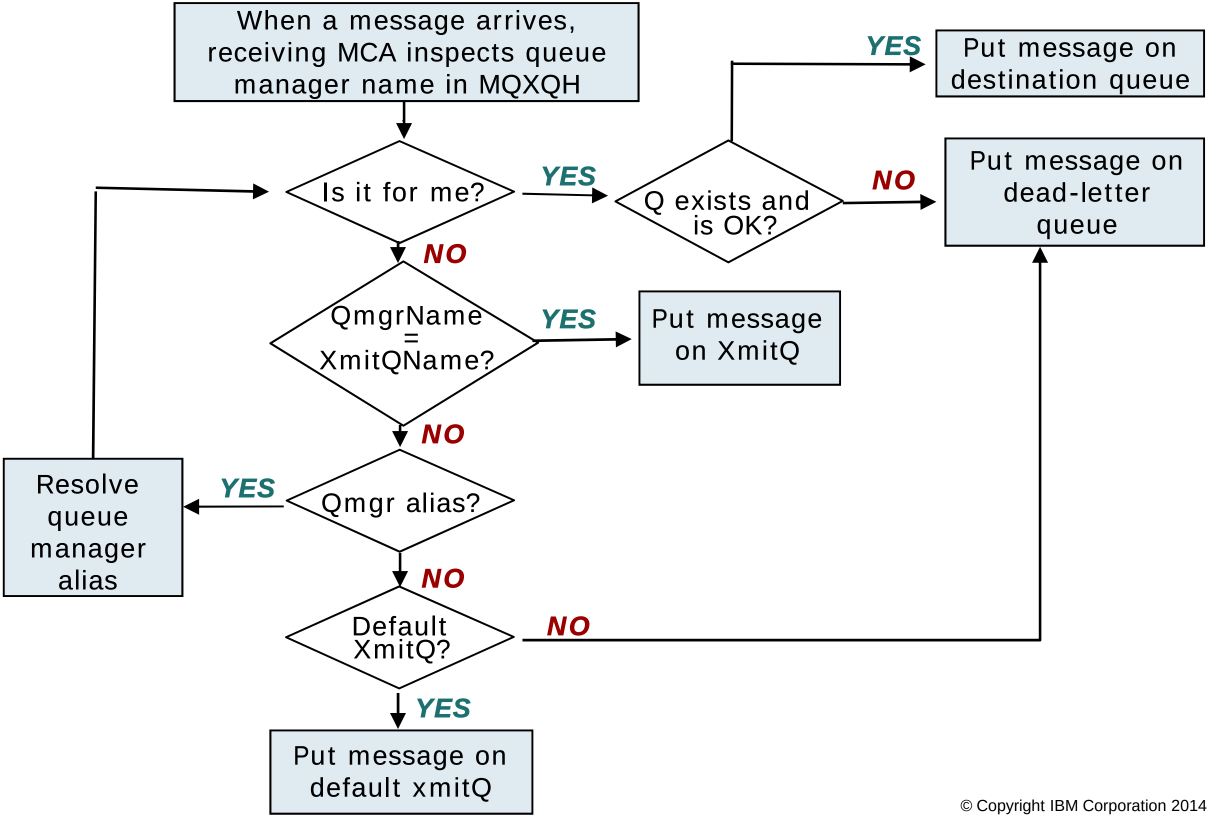
```
DEF QR(QM4)
   RQMNAME(QM4)
   XMITQ(QM3)
```

** no RNAME*

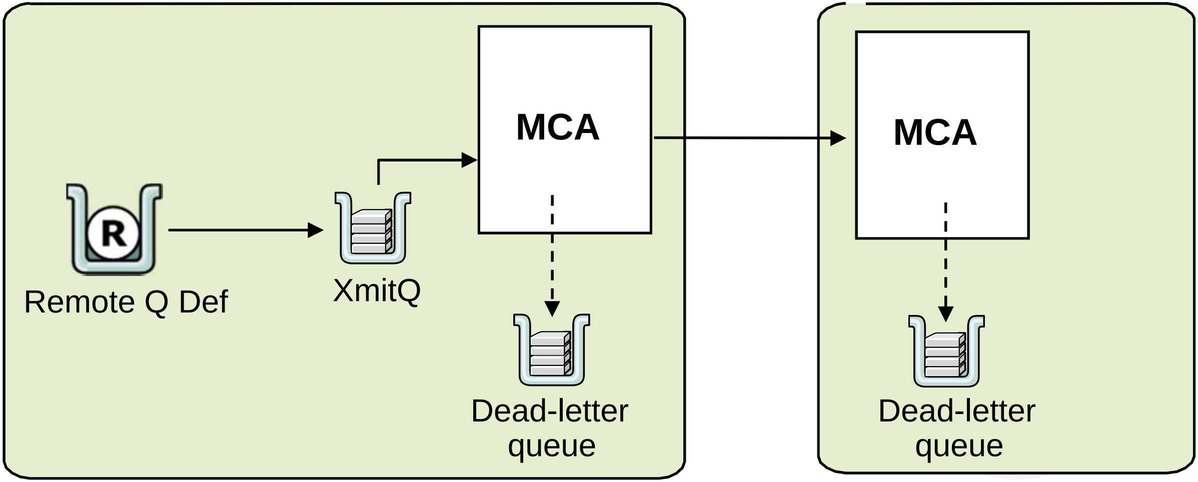
Separating message flows



When a message arrives at a queue manager

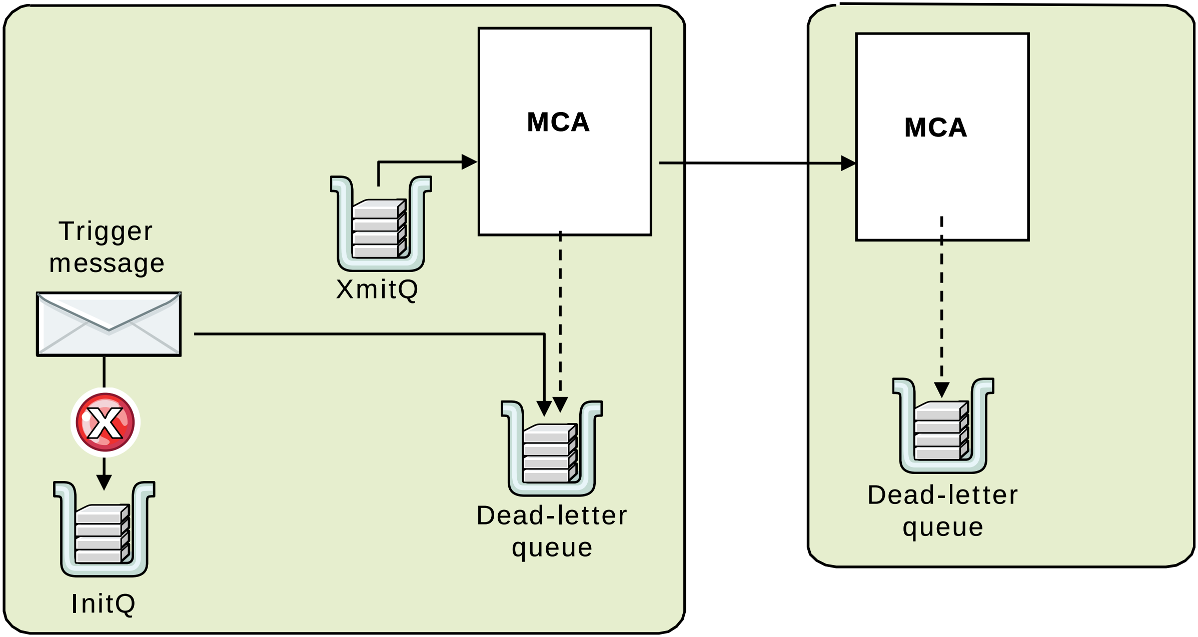

When a message arrives, receiving MCA inspects queue manager name in MQXQH
Is it for me?
YES
Q exists and is OK?
YES
Put message on destination queue
NO
Put message on dead-letter queue
NO
QmgrName = XmitQName?
YES
Put message on XmitQ
NO
Qmgr alias?
YES
Resolve queue manager alias
NO
Default XmitQ?
NO
YES
Put message on default xmitQ

Dead-letter queue (1 of 2)



- Messages that cannot be delivered are placed on the dead-letter queue
- Dead-letter queue header contains details of the destination queue name and queue manager name

Dead-letter queue (2 of 2)



- If a trigger message is created but cannot be put on the initiation queue, the trigger message is put on the dead-letter queue

Using dead-letter queues

- Create a dead-letter queue on all queue managers
- Use message retry on message channels for transient conditions
- Consider a "return to sender“ function in the application

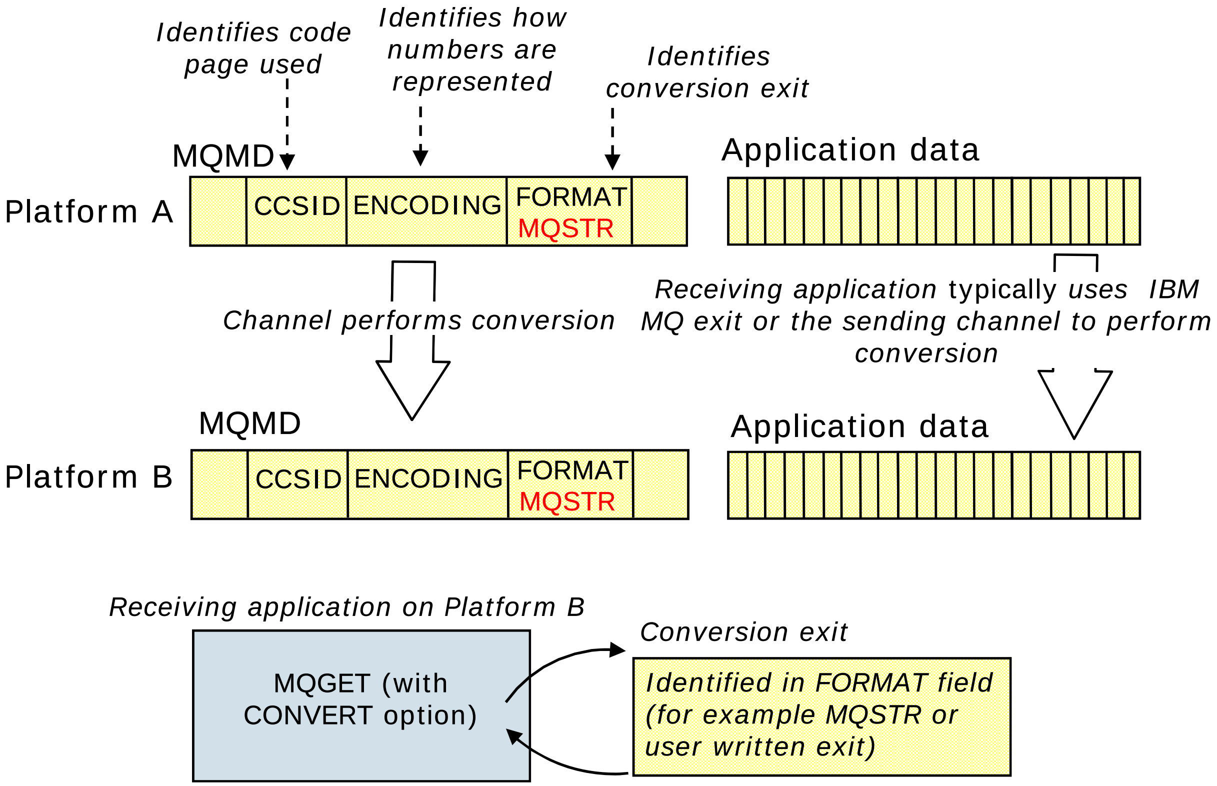
```
MQRO_PASS_MSG_ID +
MQRO_PASS_CORREL_ID +
MQRO_EXCEPTION_WITH_FULL_DATA +
MQRO_DISCARD_MSG
```

- Regularly run a routine that processes messages on the dead-letter queue to ensure that it does not become full
- WebSphere MQ Dead-letter Queue Handler utility checks messages that are on the dead-letter queue and processes them according to a set of rules that you supply to prevent an application dead letter queue from becoming full

Dead-letter header (MQDLH)

- Prefixes application message data of messages on the dead-letter queue
- Applications that put messages directly on the dead-letter queue must prefix message data with an MQDLH structure, and initialize fields with appropriate values
- Queue manager does not require that an MQDLH structure is present, or that valid values are specified for the fields
- Fields include:
  - Reason why message was put on dead-letter queue
  - Name of original destination queue and queue manager
  - Date and time message was put on queue

Data conversion



Identifies code page used
Identifies how numbers are represented
Identifies conversion exit
MQMD
Application data
Platform A
CCSID
ENCODING
FORMAT
MQSTR
Channel performs conversion
Receiving application typically uses IBM MQ exit or the sending channel to perform conversion
MQMD
Application data
Platform B
CCSID
ENCODING
FORMAT
MQSTR
Receiving application on Platform B
MQGET (with CONVERT option)
Conversion exit
Identified in FORMAT field (for example MQSTR or user written exit)

Data representation

- Messages in a heterogeneous network
  - Character fields might need translation
  - Numeric fields might need transformation
- Message descriptor accompanies every message
  - Delivered to the receiving application
  - Always converted by IBM MQ
- Application data
  - Fields in the message descriptor describe the format and representation
  - Data conversion support is available

Data representation fields in the MQMD

- **Encoding:** Representation of the numeric data in the message

  `MQENC_NATIVE`

- **CodedCharSetId:** Representation of the character data in the message

  `MQCCSI_Q_MGR`

- **Format**
  - Indicates the nature of the data in the message
  - Values starting with “MQ” are reserved for IBM MQ

Requesting application data conversion options

- Request data conversion on MQGET: **`MQGMO_CONVERT`**
  - **Encoding** and **CodedCharSetId**

    On input, requested representation of the message

    On output, what the application receives
  - Conversion performed based on **Format** field
  - A warning and message is returned in its original form, if the conversion fails

What application data conversion can be done

- Some formats are built in, and data conversion is performed by a built-in conversion routine
  - A message that is entirely character data
  - A message structure defined by IBM MQ
- User written data conversion exit is required when:
  - Application defines format of a message, not by IBM MQ
  - Message with a built-in format fails to convert
- IBM MQ utility to help write a data conversion exit

Unit summary

Having completed this unit, you should be able to:

- Diagram the connection between two queue managers by using the required components
- Configure IBM MQ channels
- Start and stop channels
- Identify channel states
- Access remote queues
- List considerations for data conversion
- Use the dead-letter queue to find messages that could not be delivered

Checkpoint questions

1. True or false: A transmission queue is required for every remotely connected queue manager.

2. True or false: A common naming convention for a transmission queue is to use the same name as the targeted remote queue manager.

3. What action prompts the channel initiator to start a sender channel?

4. What IBM MQ control command shows the current state of a channel?

Checkpoint answers

1. **True**. If the remote queue manager is not known, a default transmission queue is used.

2. **True**.

3. When a message is placed on the initiation queue of the sending queue manager.

4. Use the **`DISPLAY CHSTATUS`** command.

Exercise 7

Connecting queue managers

Exercise objectives

After completing this exercise, you should be able to:

- Design a networked IBM MQ architecture that consists of two or more interconnected queue managers
- Use IBM MQ commands to create the channels and supporting objects to implement distributed queuing
- Use the IBM MQ sample programs to test the client connection to IBM MQ