MQ Triggering

Paul Clarke

support@mqgem.com

Overview

N O T E S

- Some MQ applications that consume messages run continuously so they are always available to retrieve messages arriving on the queues.
- You may not want this with a large number of Queues and the number of arriving messages is unpredictable. In this case waiting applications are consuming systems resources even when there are no messages to retrieve.
- MQ Provides a facility that enables an application to be started automatically when there are messages available for retrieval. his facility is known as triggering.

Triggering Objectives

- What is triggering and why do you want it ?
- How does triggering work
- What can go wrong ?
  - Applications fail to trigger
  - Do they trigger too many times ?
- How to write a Trigger Monitor

Triggering Objectives

N O T E S

The objective of this session is to define triggering and show why you want to use it.

Triggering is a mechanism for starting appropriate message consumers on demand.

Delays use of memory/CPU resources until they can be productively used.

It is a tradeoff - double the number of messages processed as well as loading the application

How does it work

Certain Conditions constitute TRIGGER EVENTS.

If triggering enables for Q and Trigger Event => Message sent to Trigger Initiation Q.

Trigger Monitor reads Trigger message and Loads/starts message consumer.

Trigger Monitor is a normal MQ application consuming messages from Initiation Q

What can go Wrong

Trigger Monitor cannot start application program. Q manager cannot deliver Trigger message.

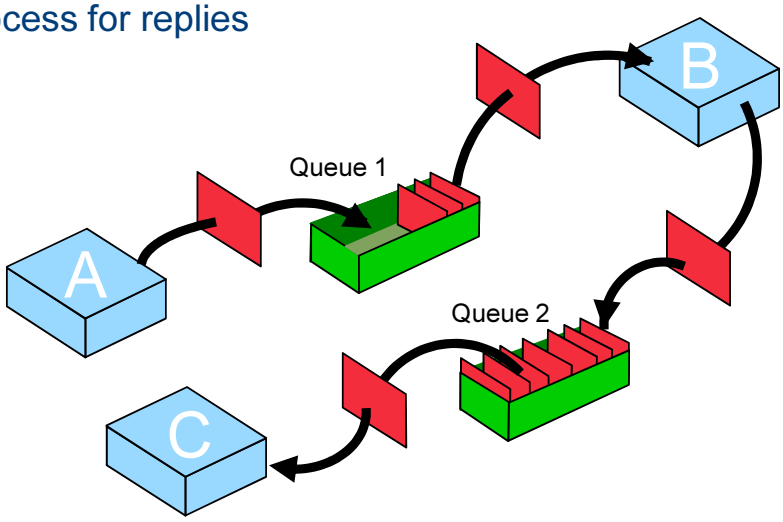
Initiation Q full. Trigger message too long. Trigger Q not open for input.=> msg written to DLQ

Symptom - messages left on application queue.

How to Write Trigger Monitors IBM ships many Trigger Monitors, others available as SupportPacs or on Forums.Customers may need to satisfy various conditions - threaded consumers, error logging

WebSphere MQ Messaging

- Time Independence
- No need for communicating programs to be active at same time
- Separate process for replies



WebSphere MQ Messaging

N O T E S

- Time independence

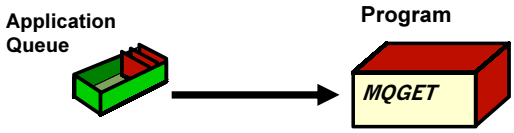
The nature of WebSphere MQ is to be able to send messages to any program at any time. However, it is not desirable to have all possible receivers of messages active, using system resources, all the time. Instead, it is much preferred to be able to have receiving applications automatically started whenever messages arrive on their respective queues.

It is for this reason that WebSphere MQ Queue Managers provide Triggering.

Capitalware's MQ Technical Conference v2.0.1.4

Triggering

- What is it ?
  - Triggering is the automatic starting of an application by the arrival of one or more messages on a queue.



- Why trigger ?
  - Unpredictable arrival of messages
  - System resources consumed when no messages

- When not to trigger?
  - Time dependence

Triggering

N O T E S

- **What is triggering ?**
- Triggering allows the user to configure the Queue Manager to associate a particular application with a particular queue and have that application started automatically when messages arrive on that queue.

- **Why trigger ?**
- Some installations have literally thousands of server queues. It is not practical to have these queues being read by a server application all the time. Not only is it wasteful of system resources such as memory and CPU but from an Administration point of view it hides the 'real' activity in the system.

- **When not to trigger ?**
- WebSphere MQ triggering does not provide any ability to configure an Application to be started based on messages arriving on a queue and the time-of-day. An end-of-day application which should run at 8pm every day, say, would not be able to use triggering and would have to rely on some other method (for example a standard job scheduler) to be started automatically.

Pseudo Triggering

- Available on
  - z/OS
  - Compaq NonStop Kernel
- MQGET (with MQGMO_SET_SIGNAL option)
  - (NO_WAIT)
  - MQRC_SIGNAL_REQUEST_ACCEPTED
- Application can continue processing until signaled
  - z/OS
    - Signal ECB posted
  - HP NonStop
    - Message put to $RECEIVE queue
- One Signal request per queue handle

Pseudo Triggering

N O T E S

- **MQGET with MQGMO_SET_SIGNAL option**

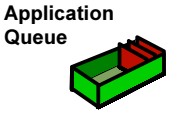
  While not strictly triggering, the MQGMO_SET_SIGNAL option does allow an application to solve the same problem as triggering in a different way. With the MQGMO_SET_SIGNAL option it is possible to have a single application monitoring for the arrival of messages on many different queues. This does mean, however, that at least one application must be running at all times. It also has the System Administration disadvantage since all target queues will be open for input.

- **How the application is Signaled**

  On z/OS, the signal is delivered by posting the Signal ECB.

  On Compaq NonStop Kernel, an inter-process communication message (IPC) is sent to the $RECEIVE queue of the process that issued the MQGET call.

Triggering Mechanism

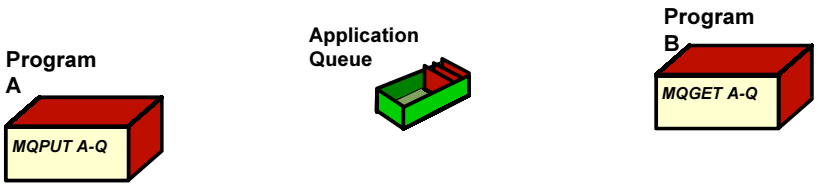


Firstly, a queue ...

Triggering Mechanism

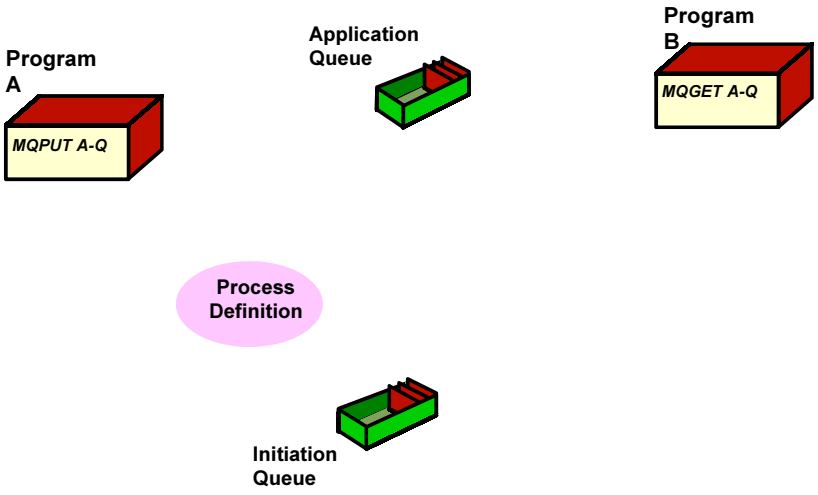
And of course, a program to write a message in the first place ...

Triggering Mechanism



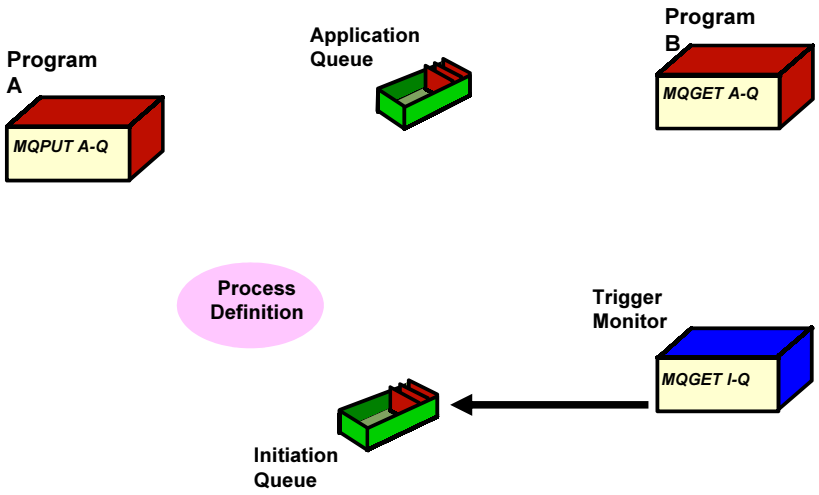
Next, a program that you want to be started when a message arrives on that queue ...

Triggering Mechanism



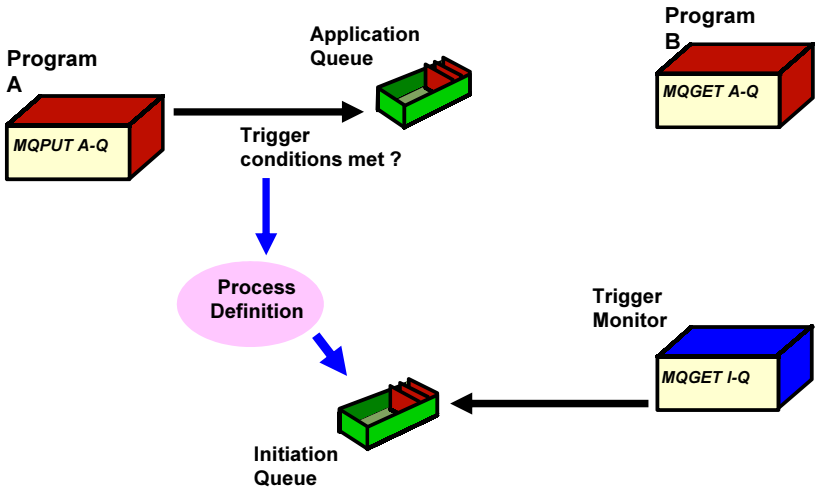
Next, a process definition and an initiation queue ...

Triggering Mechanism



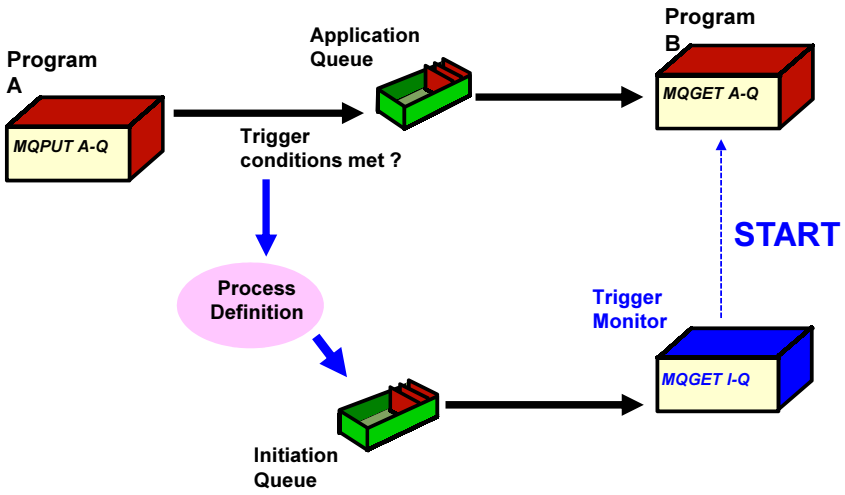
Next, a Trigger Monitor program waits for messages on the initiation queue.

Triggering Mechanism



Trigger message that contains values from application queue and process definition is generated by Queue Manager.

Triggering Mechanism



Trigger message received by Trigger Monitor program.
Trigger Monitor starts program which reads original message.

Triggering Mechanism

N O T E S

- **Trigger Conditions**
  There are a number of conditions the Queue Manager must check before generating a trigger message, for example, it checks whether the application queue is already open for input. If it is then there's little point generating a trigger message since an application is already processing the queue.
- **Process Object**
  The Queue Manager ultimately needs a definition of the program to be started, for example its name and start parameters. These values are stored in a Process object within the Queue Manager. It is associated with a particular Application Queue by the process objects name being identified in the Queue definition.
- **Initiation Queue**
  This is just a local queue where the Queue Manager can put trigger messages. The name of the initiation queue to use is another of the local Application Queue's attributes. Many application queues can, and probably will, identify the same initiation queue.
- **Trigger Monitor**
  This is a program whose sole responsibility is to monitor the initiation queue for trigger messages and start the application identified in the trigger message. The trigger monitor program is generally started at the time the Queue Manager itself is started. A single trigger monitor is sufficient for any number of applications, however, it is common to have different trigger monitors for different environments. For example (batch and CICS). WebSphere MQ provides a number of trigger monitors with the products or it is quite feasible to write your own.

Base Trigger Conditions

- Trigger Control Enabled
- Trigger Types
  - FIRST
    - By far the most common triggering mode
    - Trigger Message generated as queue depth goes from 0 to 1
  - DEPTH
    - Trigger Message generated when queue depth reaches trigger depth
  - EVERY
    - Trigger Message generated for each application message
  - NONE
    - Triggering turned off
- Initiation Queue Identified in Application Queue definition

Further Triggering Conditions

- Process Object Identified in Application Queue definition
- Application Queue
  - Not already open for input (FIRST and DEPTH)
  - Is MQGET enabled
- Message priority >= Trigger priority
  - Only messages greater than or equal to the queue's trigger priority will generate a trigger message
  - e.g. On a FIFO queue, triggering will only ever occur if default priority is >= trigger priority
- Initiation Queue
  - MQPUT & MQGET enabled
  - Open for input. i.e. Trigger Monitor is running!

Further Triggering Conditions

N O T E S

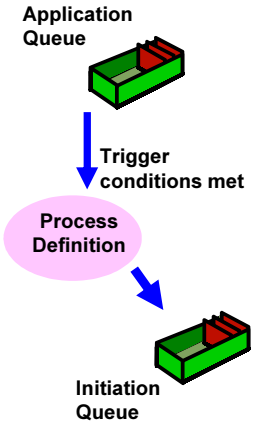
- **FIFO queues**
  Messages put to a queue defined with a delivery sequence of FIFO are put with the queues default priority. Therefore, to trigger a FIFO queue, the queue's default priority must be set to greater than or equal to the queue's trigger message priority.

Capitalware's MQ Technical Conference v2.0.1.4

Triggering Configuration

```
DEFINE  QLOCAL(appl.q)
        TRIGGER
        TRIGTYPE(first)
        TRIGDATA(<char 64>)
        TRIGMPRI(0)
        PROCESS(proc1)
        INITQ(initq)

DEFINE  PROCESS(proc1)
        APPLICID('c:/progB')
        APPLTYPE(def)
        ENVDATA(<char 128>)
        USERDATA(<char 128>)
```



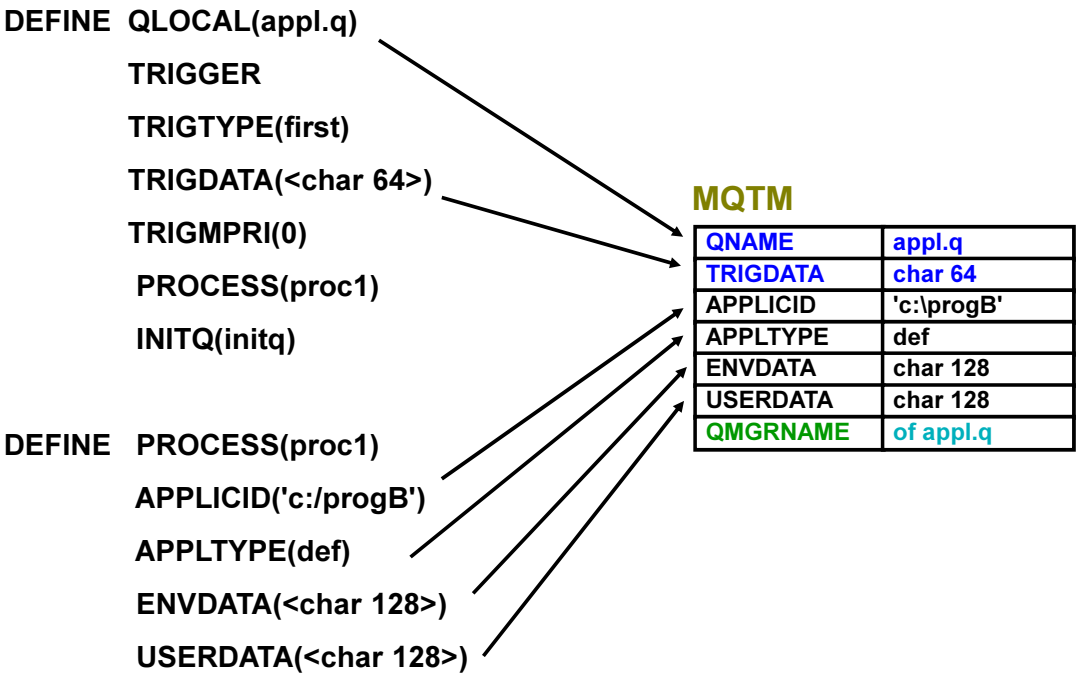
Triggering Configuration

N O T E S

- **Initiation Queue**
  It is normal to have one initiation queue per environment, in other words, all applications to be started in that environment share the same initiation queue, and hence the same trigger monitor. However, if required, as many trigger monitors as necessary can be used.

- **Process Object**
  While it is normal to have a different process object defined for each local queue, it is perfectly possible to have the same process object definition identified in many queue definitions. This is useful when the same application is used to process many different queues.

Trigger Message



Trigger Message

N O T E S

- **Trigger Message**
  The content of the trigger message is generated from attributes of the application queue itself and its process object.

QNAME	Name of queue being triggered.
TRIGDATA	Parameters to application on a per queue basis.
APPLICID	Application name.
APPLTYPE	Identifies the type of application to the trigger monitor.
ENVDATA	Used by trigger monitor to control the way the application is started. eg.'&' used by IBM Unix Trigger Monitor to indicate background job
USERDATA	Parameters to application on a per process basis
QMGRNAME	Local Queue Manager name.

Trigger Message Attributes

- Trigger Messages always non-persistent
  - Reduces overhead
  - Message regenerated if required
- Unit of Work
  - Trigger Messages generated in same unit of work as message causing trigger.
  - Trigger Message committed even if application backs out.
    - Can lead to trigger message with no application message

Trigger Message Attributes

NOTES

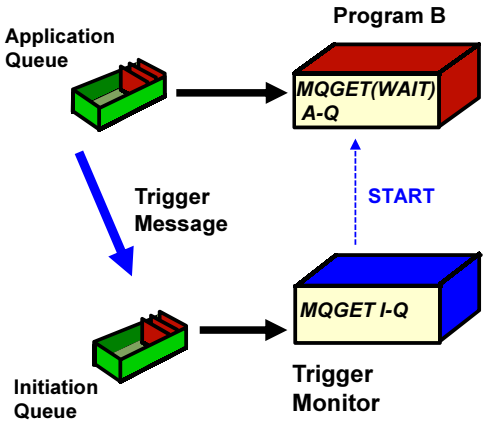
- Unit of Work

The most common form of triggering (TRIGTYPE = FIRST) only triggers on the depth of the queue going from 0 to 1. Since WebSphere MQ queue depth counts the uncommitted messages it follows that an application putting a message inside a unit of work will prevent any other application putting to the same queue from generating trigger messages. If the original application backs out its unit of work then there may still be other messages on the queue which need to be processed. By always committing the trigger message we ensure that the worst that will happen is that a trigger message is generated even when there are no messages on the queue. It is therefore necessary to code triggered applications so that they do not mind if they get started even when there are no messages to process.

Capitalware's MQ Technical Conference v2.0.1.4

Triggered Application Processing

- Parse parameters
  - May be initiator dependent
- Open input queue (once)
- Do MQGET(WAIT) loop
- No message is not an error
- Allow for no messages
- Only end after no messages have arrived for some time
  - Wait time affects performance



Triggered Application Processing

NOTES

- MQGET(WAIT)
  It is strongly recommended that any triggered application waits around for some time after emptying the application queue. This is to prevent excessive starting of the application as each message arrives on the queue. The ideal situation is to have the application already running and ready to process messages at the time the messages are put to the queue. However, you don't want to set the wait interval too large otherwise it defeats the whole point of triggering.

- Trigger Type
  The exact processing performed by the application may depend on the trigger monitor used and the type of triggering. The simplest and most common form of triggering is TRIGTYPE(FIRST). However, if TRIGTYPE(EVERY) or TRIGTYPE(DEPTH) is used then the application must be aware of this and perform additional processing explained later.

Capitalware's MQ Technical Conference v2.0.1.4

Trigger Message Generation

- Trigger Messages will be generated at the following times, provided the triggering conditions are met
  - When a message is put to an application queue
  - When an application queue is closed
  - When an initiation queue is opened for input
  - When the application queue trigger type attribute is changed
  - When the Queue Manager Trigger Interval is reached

Trigger Message Generation

NOTES

In all the cases below, trigger messages will only be generated if the previously described trigger conditions are met.

- When a message is put to an application queue
  Normal operation
- When an application queue is closed
  By generating a trigger message when the application queue is closed the application need not worry about a message being put to a target queue between it's last MQGET() call and it issuing an MQCLOSE().
- When an initiation queue is opened for input
  Generating messages at the time the trigger monitor is started means that when the Queue Manager is started with messages already on the queues the act of starting the trigger monitors will ensure that the appropriate applications will get started to process those messages.
- When the application queue trigger type attribute is changed
  This allows an operator to temporarily suspend triggering of an application and subsequently restart it.

What can go wrong?

- Application not started
  - Check all the triggering rules
  - Initiator failing to start application process
  - Application queue already open
  - Initiator is starting applications in foreground
    - Indicated by only one trigger working at a time
- Application started too much
  - Application is closing the application queue too early

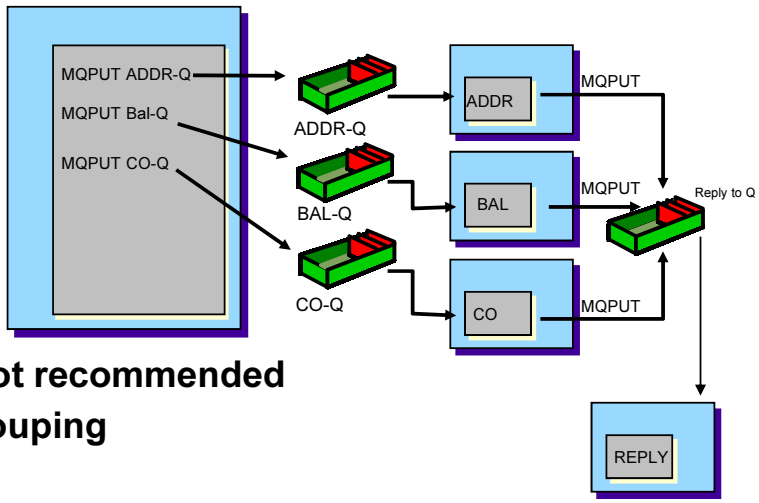
What can go wrong?

NOTES

- Triggering Rules
  For the sake of brevity this presentation lists only the major triggering rules. A complete list of rules required for triggering is given in the MQ documentation. Please see:
  http://www.ibm.com/support/knowledgecenter/SSFKSJ_8.0.0/com.ibm.mq.dev.doc/q026930_.htm
- Initiator failing to start the process
  This can be a difficult problem to solve. The supplied trigger monitors do not log error messages if they fail to start the application, however, system error messages are written to stdout. It is worth running the trigger monitor in a foreground session or piping its output to a file. If all else fails an MQ trace will show explicitly whether a trigger message is being generated.

TRIGTYPE(DEPTH)

TRIGTYPE(DEPTH) TRIGDPTH(3)



- **This type of processing is not recommended**
  - **Better to use message grouping**
- **Instead use for batch, non-performance sensitive processing**
  - **Need to code alternative start mechanisms**
- **Note that TRIGTYPE(DEPTH) turns off triggering**
  - **Application must re-enable**

TRIGTYPE(DEPTH)

N O T E S

- TRIGTYPE(DEPTH) can be useful to cause the start of the application only when a predetermined number of messages have arrived on the queue. There are a number of problems with using this mode of operation however :-
  - Server queues can only be processed by a single application. No parallelism can be exploited.
  - A missing message can lead to orphaned messages on the queue since the application will never get started.
  - The application must 'know' it's being started with TRIGTYPE(DEPTH) and re-enable triggering for the queue before it terminates.
  - Be aware that the Queue Manager has no notion of the associativity of the messages on the queue, only the number of messages is looked at. This can lead to false triggers if the queue contains orphaned messages or is used in parallel by multiple requesters.
  - If the queue is a REPLY queue it should probably be a dynamic queue to ensure that it is only being used for a single business transaction.
- Message Segmentation/Grouping (V5.0)
  In most cases Message Segmentation is a better way of solving the kind of problems TRIGTYPE(DEPTH) was used for. With segmentation and grouping of messages an application can start work on a group of messages knowing that all messages are available.

TRIGTYPE(EVERY)

- Trigger message written for every message
  - Current Queue Depth is irrelevant
  - Intended for transactions that process only one message
  - Only trigger type which will start multiple instances of the application
- Only one message if Trigger Monitor is restarted
  - Queue Manager will not generate a trigger message for each message on the queue.
  - Applications must have an alternative way of processing all messages on the queue at startup.
- Trigger FIRST is usually easier to get right
  - Only use TRIGTYPE(EVERY) if you have to !

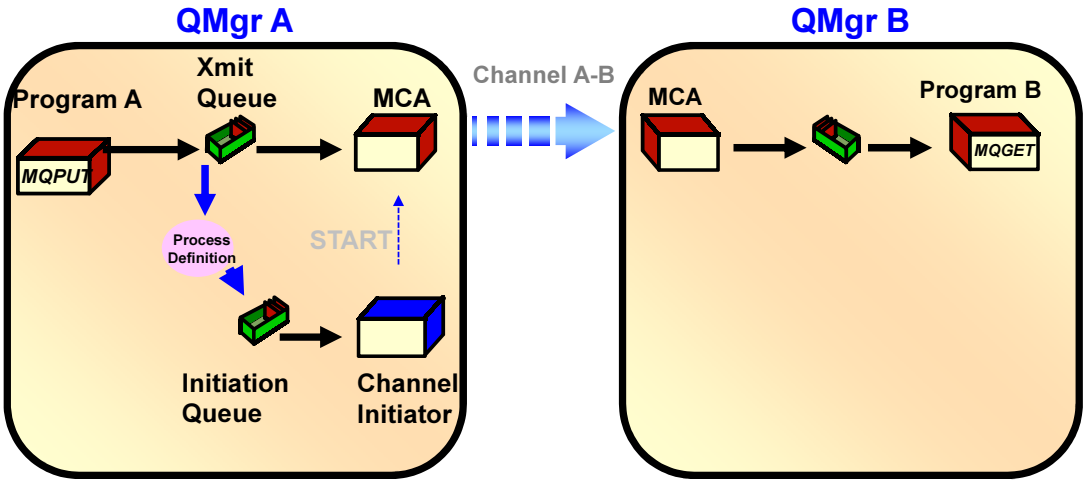
TRIGTYPE(EVERY)

N O T E S

- TRIGTYPE(EVERY)
  This trigger type will cause the Queue Manager to generate a trigger message every time a message is put to an application queue. This is very inefficient since it causes much more processing in the queue manager and is generally only used for legacy applications which can't process more than one message at a time.
- There is another use which is fairly common and that is load balancing. As things stand the Queue Manager will not load balance across a number of applications. The trigger monitor will start an instance of an application and the Queue Manager will stop generating trigger messages. However, with TRIGTYPE(EVERY) the Queue Manager will continue to generate trigger messages. It is therefore possible to write a trigger monitor which, upon receiving a trigger message, will issue a query against the application queue and look at the current depth and the number of applications processing the queue. It can then start a new instance of the application if it appears there are too few applications for the number of messages on the queue.
  This kind of processing is fairly common in CICS installations.
- Channels
  Do not expect to be triggered with TRIGTYPE(EVERY).
- Only set transmission queues to TRIGTYPE(FIRST) or TRIGTYPE(DEPTH).

Capitalware's MQ Technical Conference v2.0.1.4

Triggered Channels




```
DEFINE  QLOCAL(QMgrB)
        USAGE(XMITQ)
        TRIGTYPE(first)
        INITQ(channel-initq)
        TRIGDATA(channel a-b)   (Optional)
        PROCESS(channel-proc) (Optional)
```

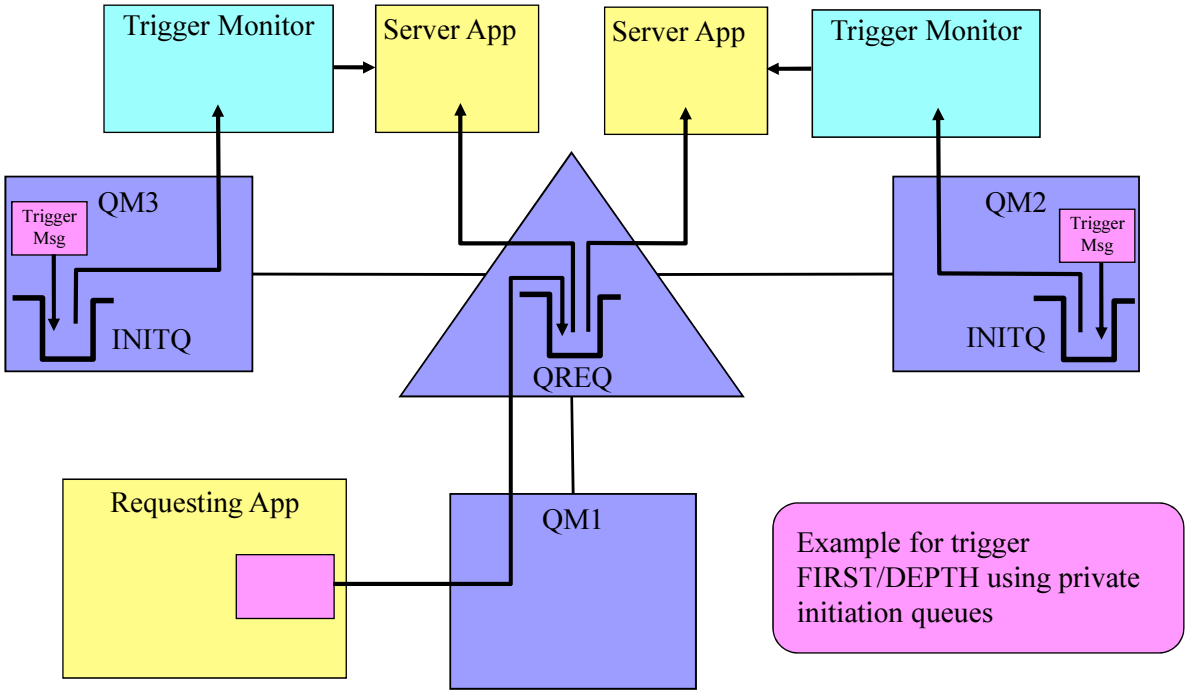
```
DEFINE   PROCESS(channel-proc)
         USERDATA(channel a-b) (Optional)
```


Triggered Channels

N O T E S

- Perhaps the first use of a triggered application that's used in any installation is for automatically starting the channels. Because of the special processing requirements of channels (e.g.. we want to be able to retry connections) we use a special form of a trigger monitor called the Channel Initiator. However, as far as the Queue Manager is concerned the channel is just like any other triggered application. The only concession, done in V5.0, was to remove the requirement that Transmission Queues need to have a process object associated with them for triggering to occur.
- As with all good triggered applications the Channels allow the user to customize the time of inactivity before they end. This is known as the Disconnect Interval. It allows an Administrator to customize the starting of channels in accordance with the message arrival rates.
- Normally all channels are handled by a single Channel Initiator. However, on Distributed platforms it is possible to have multiple channel initiators.
- When a channel is manually stopped or in RETRY it will set the transmission queue to NOTRIGGER and GET(DISABLED). This temporarily suspends triggering to prevent unwanted starts.
- It should be noted that cluster channels are not started by triggering. The channel initiator is still involved but the Queue Manager makes explicit start requests for these types of channels.
- On z/OS, if you choose to use a process definition to trigger start channels, then you must also set APPLICID("CSQX START") and APPLTYPE(MVS). Refer to the description of DEFINE PROCESS in the WebSphere MQ Command Reference (SC34-6055-00).

Triggering in a Queue Sharing Group (QSG)

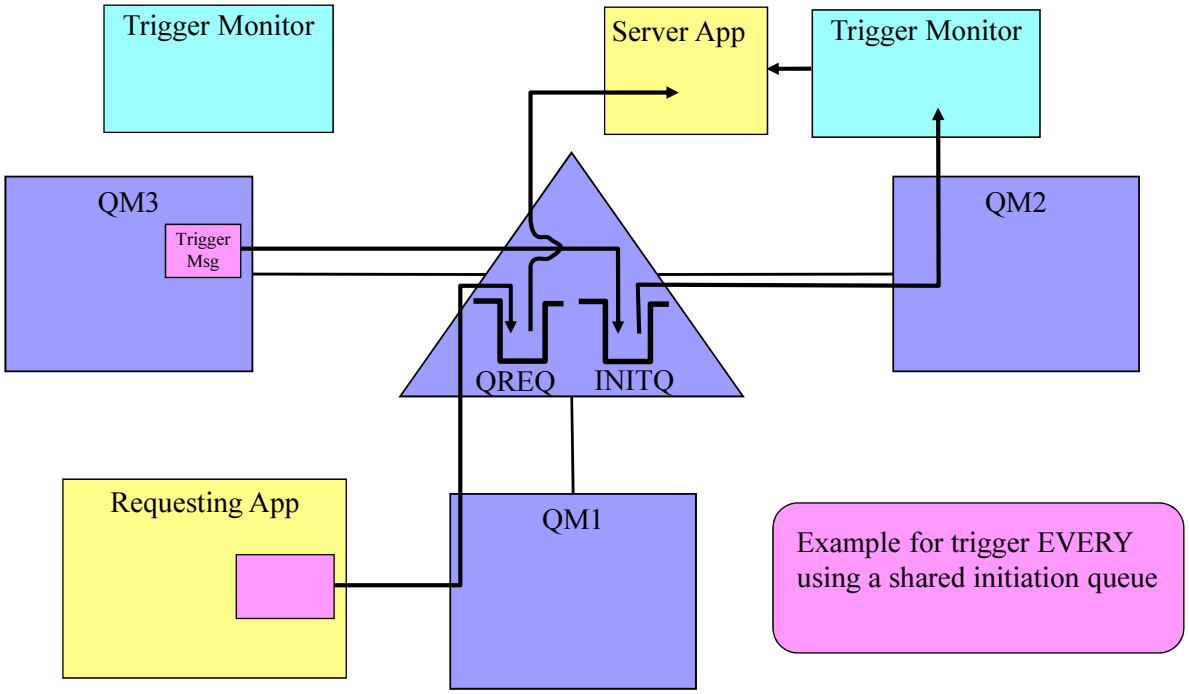


Triggering in a Queue Sharing Group (QSG)

N O T E S

- For shared queues, triggering is only performed on committed messages.
- When performing triggering based on FIRST or DEPTH, the QMgr uses the Coupling Facility list monitoring services to know when a new message has arrived on a particular queue, and hence a trigger message needs to be generated.
- The generation of trigger messages is performed by each QMgr independently of the other QMgrs in the QSG. The initiation queue being private or shared does not affect how the QMgrs generate the trigger messages. Assuming the normal triggering requirements are met then the QMgr will create a trigger message and place it onto the initiation queue. This means that multiple trigger messages may be created, up to N (where N is the number of QMgrs in the QSG).
- In the example, two QMgrs have private initiation queues. This means that a server application will be started on each of these QMgrs. Depending on the number of messages on the triggered queue, there may be a “race” condition between the servers that are started and some many get MQRC_NO_MSG_AVAILABLE. The server application should be able to cope with this situation.

Triggering EVERY in a Queue Sharing Group (QSG)



Triggering EVERY in a Queue Sharing Group (QSG)

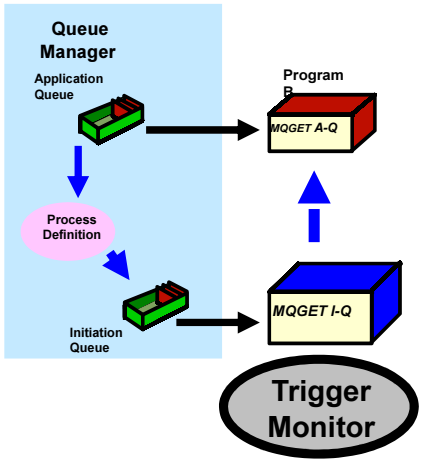
N O T E S

- When performing trigger EVERY the QMgr where the put is taking place nominates a QMgr in the QSG to produce the trigger message.
- As shown in the diagram, a single trigger message is created by the nominated QMgr. In this case a trigger monitor connected to a different QMgr retrieves this trigger message and starts the server application.
- It is advisable that when using trigger EVERY, the initiation queue should be a shared queue. If it were a private queue, then it might be possible for the trigger message to be placed on the initiation queue, but either the trigger monitor or the QMgr to fail before it gets processed, meaning that the message that caused the trigger to be created may remain unprocessed.

Why write your own trigger monitor?

User Trigger Monitor:

- Invoke programs differently
  - e.g. Start Application as a thread
- Extra features
  - e.g. RETRY or Time Dependence
- Nonstandard error processing
- Nonstandard parameters
- Different Application Environment
  - e.g. IMS BMP



Why write your own trigger monitor?

NOTES

- Supplied Monitors
  IBM provides a whole range of monitors which cover a wide range of application environments. Most are provided with the products themselves but there are also ones available on the WebSphere MQ SupportPac site (http://www.ibm.com/software/integration/support/supportpacs/)
- If a suitable monitor does not appear to be available it is certainly worth asking on the various MQ fora since the sharing of this type of information is commonplace.
- In most cases, monitor programs are provided complete with source code. This makes it relatively easy to modify the behaviour to suit installation requirements.

Capitalware's MQ Technical Conference v2.0.1.4

Trigger Monitor - Basic Design

MQCONN

MQOPEN Initiation Queue

while(Always)

{

MQGET message (Unlimited Wait)

Reject anything that is not a trigger message (DLQ)

Build application parameters (e.g. MQTMC2)

"run" program/transaction, passing parameters

}

MQCLOSE

MQDISC

Trigger Monitor - Basic Design

N O T E S

- MQGET
  - Issue MQGET(WAIT_UNLIMITED).
  - Use MQGMO_FAIL_IF_QUIESCING so that the monitor terminates immediately upon Queue Manager termination.
  - Issue MQGET outside of syncpoint.
- Message rejection
  If the monitor receives a message other than one it is expecting it should, as with any other MQI Application, put the message to the DLQ.
- Application Parameters
  The standard parameter to a triggered application is the MQTMC2 structure. This contains the entire content of the trigger message (MQTM) in character format. Use MQTMC2 unless you actually need a different format.
- Error logging
  The monitor should report errors if it fails to start the application. This makes problem determination far easier.
- Triggering
  In general monitors should run when ever the Queue Manager is running. Note that trigger monitors themselves can not be triggered. They should be started manually or at Queue Manager start-up

Trigger Message Test

- Test for valid Trigger Message

```
                                        /* Test format      */
if (memcmp(trig->StrucId, MQTM_STRUC_ID, 4)||
    memcmp(md.Format, MQFMT_TRIGGER, MQ_FORMAT_LENGTH))
{
                                        /* Build DLQ Header */
  dlh.Reason          = MQFB_TM_ERROR;
  dlh.Encoding        = md.Encoding;
  dlh.CodedCharSetId  = md.CodedCharSetId;
  dlh.PutApplType     = MQAT_DEFAULT;
  memcpy(dlh.Format, md.Format, MQ_FORMAT_LENGTH);
  memcpy(dlh.PutApplName, Applicid, MQ_PUT_APPL_NAME_LENGTH);
  memcpy((dlh+1),msg,msglen);

  MQPUT(...dlh,...);
  continue;
}
```

Trigger Message Test

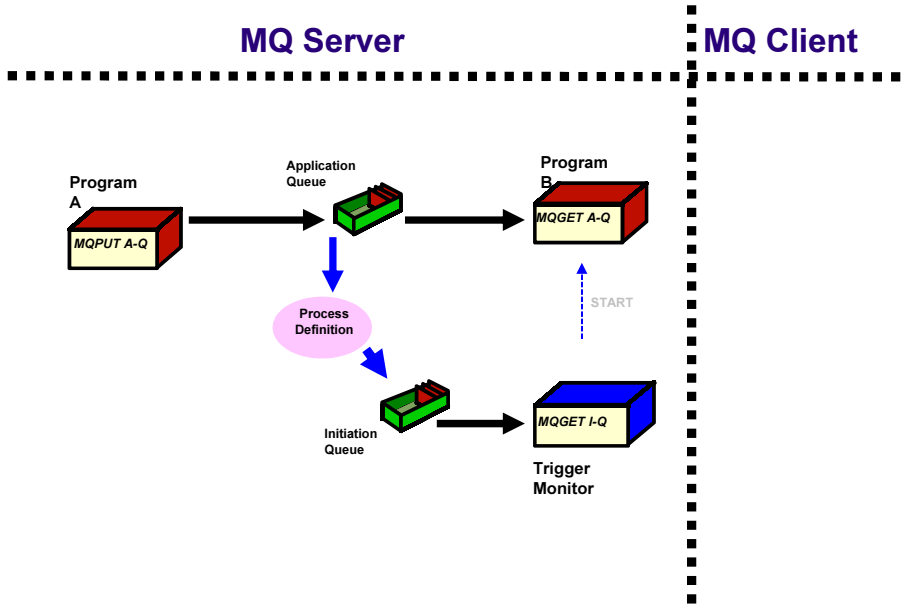
N O T E S

- **Testing for trigger message**
  To be safe test both the Format field in the message descriptor and the first few bytes of the message for the trigger eyecatcher.

  Some monitors also check the trigger message version number however, as a general rule, subsequent structure versions are super-sets of current versions.

- **Putting to DLQ**
  The Dead Letter Queue name can be retrieved by issuing an MQINQ against the Queue Manager object.

Triggering - MQ Clients

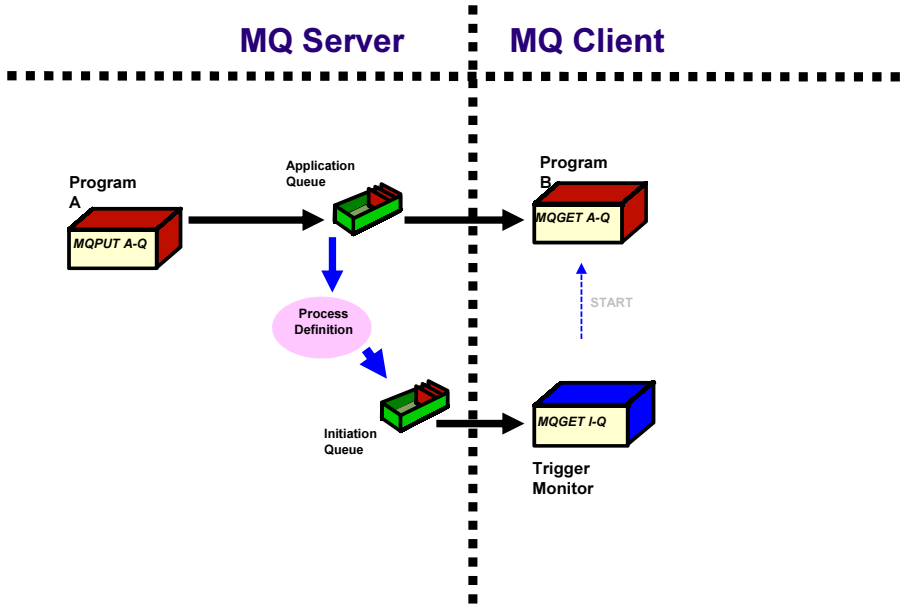


Triggering – MQ Clients

N O T E S

- Here is the picture for the normal MQ applications for triggering. Notice that both the Trigger Monitor and the triggered program are just MQ applications. This means that they can both be run as clients as shown on the following chart.

Triggering - MQ Clients



Triggering – MQ Clients

N O T E S

- In order to trigger an application that runs as a client the program start request must be issued on the client machine.
- With the IBM supplied trigger monitors this is normally done by also running the trigger monitor on the client machine as a client program. A client version of the standard trigger monitor RUNMQTRM is supplied and is called RUNMQTRC.
- Since trigger monitors are just normal MQI applications a client version of any monitor can be created merely by linking with the client rather than the Queue Manager library.
- The disadvantage in using a client trigger monitor is that since the trigger monitor should be running for the whole life of the Queue Manager, it will use a client session for the whole life of the Queue Manager.
- For some applications it is possible to use a more complicated start command (e.g. REXEC) to start a program on a remote machine. Using this method would obviously only require a local trigger monitor.

Summary

- WebSphere MQ Triggering
  - When to use it
  - Configuration
- Trigger Conditions
  - Necessary conditions to make it work
  - What can go wrong
  - Channel triggering
  - DEPTH and EVERY
- Trigger Monitors
  - Why and how to write a user trigger monitor

Thank-you – Any questions?