Efficiently Accessing MQ Messages from IMS Applications

Session 16096

Steve Nathan – snathan@us.ibm.com

#SHAREorg

SHARE is an independent volunteer-run information technology association that provides **education**, professional **networking** and industry **influence**.

Disclaimer

IMS Application Access to MQ Messages

- **For Your Information**
  - As of MQ Version 8, WebSphere MQ is now called MQ – again

IMS Application Access to MQ Messages

- **Acknowledgements**
  - I would like to thank Luigi Certorelli for helping me to analyze all the ways that IMS applications can access MQ messages.
  - I would like to thank Pete Sadler for his great comments on how to improve this presentation.
  - I would like to thank Suzie Wendler and Ken Blackman for their suggestion of the Enhanced MQ IMS Queue Monitor.

IMS Application Access to MQ Messages

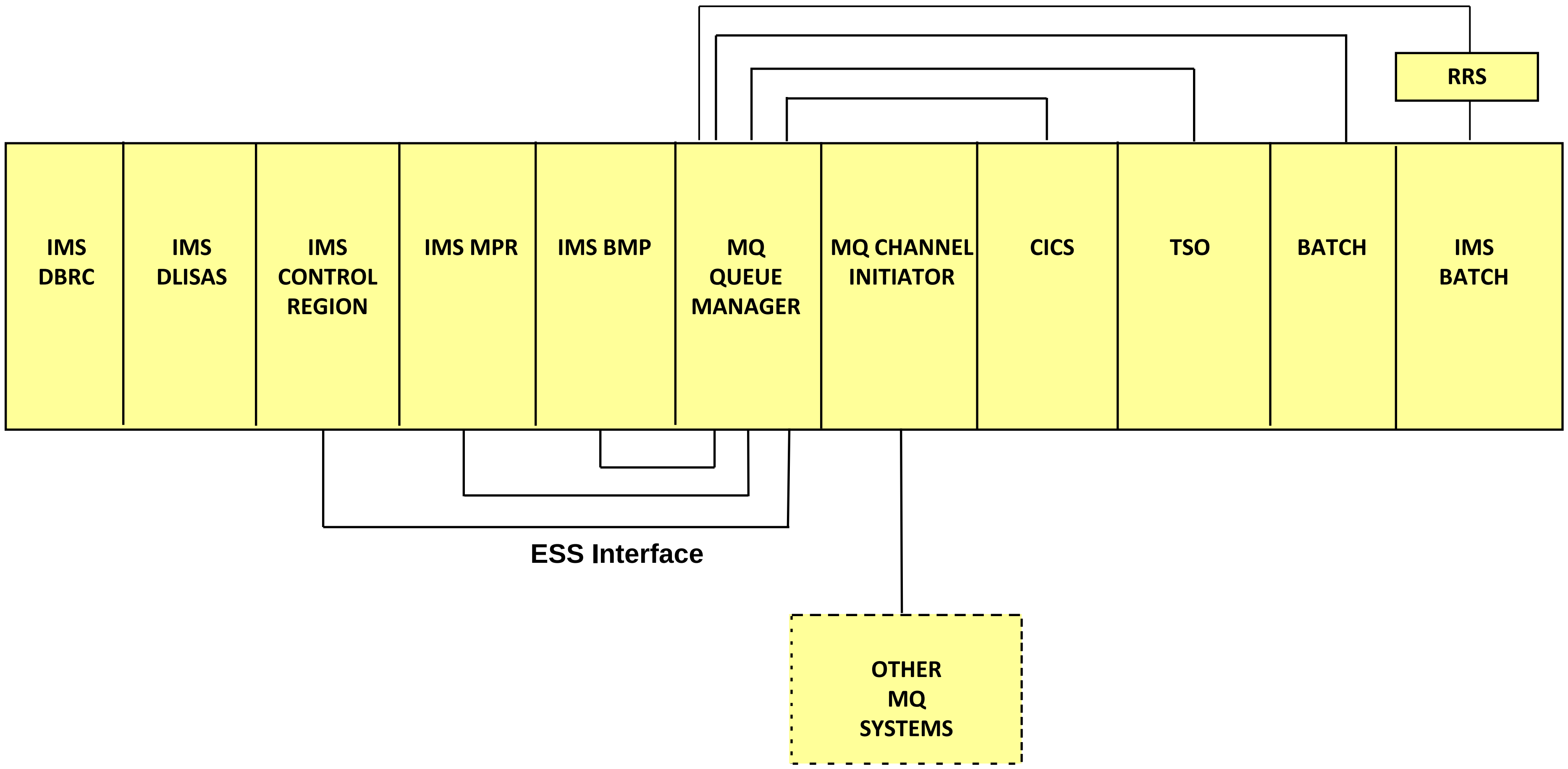
- **IMS applications can access MQ messages in two ways:**
  - 1. The IMS application uses the MQ API to Get and Put messages with syncpoint coordination with IMS
    - IMS BMP MPP IFP (not JMP or JBP - until IMS 13)
      - Requires connecting MQ to IMS via ESS
      - Link the application program with the MQ IMS stub (CSQQSTUB)
    - MQ messages can be inserted to the IMS Message Queue by an application program (BMP/MPP)
      - Could be a Trigger message (MQ IMS BMP Trigger Monitor)
      - Could be the real Message
    - IMS Batch
      - No ESS interface
      - Syncpoint coordination requires RRS
      - Link the application program with the MQ two-phase commit batch stub
        - CSQBRRSI or CSQBRSTB+ATRSCSS

IMS Application Access to MQ Messages

- **IMS applications can access MQ messages in two ways:**
  - **2.** The MQ IMS Bridge puts the message on the IMS Message Queue via OTMA
    - The MQ IMS Bridge is code in the MQ Queue Manager
    - Does not require connecting MQ to IMS via ESS
      - But the ESS connection could also exist for programs using the MQ API
    - Requires OTMA configuration in the MQ CSQZPARM

Connecting MQ and IMS via ESS

- **MQ for z/OS attaches to IMS just like DB2 using the external subsystem (ESS) (ESAF) interface**



Connecting MQ and IMS via ESS

- **Place the MQ authorized library (HLQ.SCSQAUTH) in the IMS control region and dependent region DFSESL concatenations**
- **Copy module CSQQDEFV from HLQ.SCSQASMS to be customized, assembled, and linked into a library in the IMS dependent region(s) STEPLIB concatenation**
  - IMS ABENDU3041 if it is not there
  - Old documentation said IMS Control Region STEPLIB and authorized but this in incorrect
  - **Match to the LIT's in the IMS SSM members (next foil)**

```
CSQQDEFV CSECT
          CSQQDEFX NAME=CSQ1,LIT=LIT1,TYPE=DEFAULT
          CSQQDEFX NAME=CSQ3,LIT=LIT2
          CSQQDEFX TYPE=END
```

Connecting MQ and IMS via ESS

- **Define MQ to IMS by adding SSM information to the IMS PROCLIB (member name *IMID*xxxx)**

  FORMAT: SSN,LIT,ESMT,RTT,REO,CRC
  - SSN: Subsystem Name - MQ subsystem
  - LIT: Language Interface Token - **From CSQQDEFV**
  - ESMT: External Subsystem Module Table - “CSQQESMT”
  - RTT: Resource Translation Table - Not Used by MQ
  - REO: Region Error Option - “R”, “Q”, or “A” (pick any one)
  - CRC: Subsystem Recognition Character - Not Used by MQ
    - The /SSR command is not supported
  - The keyword format of this PROCLIB member is not supported for MQ

Connecting MQ and IMS via ESS

- **Subsystem Connection**

```
/DIS SUBSYS ALL

SUBSYS    CRC  REGID  PROGRAM  LTERM      STATUS
CSQ3      !                               CONN
CSQ1      <                               CONN
DB2R      =                               CONN
                 1                        CONN
                 5                        CONN
```

Using the MQ API with IMS Applications

- **MQ application stubs**
  - An application program must be linked with a “stub” module in order to use the MQ API
  - There are three possible “stubs” that can be used in IMS applications

1. CSQQSTUB
   - IMS stub
   - MQ knows the application is running in an IMS environment
   - Provides two-phase commit for IMS and MQ API calls
     - IMS is the syncpoint coordinator
   - Not for IMS Batch (DLI/DBB)
2. CSQBSTUB
   - Batch stub
   - MQ does not know the application is running in an IMS environment
   - There is no two-phase commit with IMS
   - Can be used for online processing – more later

Using the MQ API with IMS Applications

- **MQ application stubs (continued)**
  - An application program must be linked with a “stub” module in order to use the MQ API
  - There are three possible “stubs” that can be used in IMS applications

3. CSQBRSTB
   - Batch two-phase commit stub
   - Can only be used in IMS batch jobs
   - Requires RRS

Using the MQ API with IMS Applications

- **Calls to MQ, IMS and DB2 can be made within the same unit of work (UOW)**
  - MQ API calls
  - IMS IOPCB calls
  - IMS ALTPCB calls
  - IMS database calls
  - DB2 calls

Using the MQ API with IMS Applications

- **IMS and MQ Units of Work**
  - An IMS commit is also an MQ and DB2 commit
    - SYNC, CHKP, GU to IOPCB (MODE=SNGL), normal program termination
  - An IMS backout (ROLB) is also an MQ and DB2 backout
  - Any IMS abend is also an MQ and DB2 backout
    - ROLL, miscellaneous abends

Using the MQ API with IMS Applications

- **At normal syncpoint....**
  - IMS input message is dequeued
  - IMS NON-EXPRESS output messages are sent
  - IMS EXPRESS output messages have already been sent
  - IMS database updates are committed
  - DB2 updates are committed

Using the MQ API with IMS Applications

- **At normal syncpoint....**
  - MQ input messages marked with SYNCPOINT, or MARK_SKIP BACKOUT are dequeued
  - MQ input messages marked with NO_SYNCPOINT have already been dequeued
  - MQ output messages marked with SYNCPOINT are sent
  - MQ output messages marked with NO_SYNCPOINT have already been sent
  - If the IMS application is message driven (BMP or MPP) the MQ connection handle is closed by MQ for security reasons
    - Connection security is by Userid
    - Each message can be from a different Userid
    - (More later)

Using the MQ API with IMS Applications

- **At abnormal termination or ROLx....**
  - IMS input message is dequeued
    - IMS has Non-Discardable Message Exit
  - IMS NON-EXPRESS output messages are discarded
  - IMS EXPRESS output messages have already been sent
  - IMS database updates are backed out
  - DB2 updates are backed out

Using the MQ API with IMS Applications

- **At abnormal termination or ROLx....**
  - MQ input messages marked with SYNCPOINT are ***re-queued***
  - MQ input messages marked with NO_SYNCPOINT have already been dequeued
  - MQ input messages marked with MARK_SKIP_BACKOUT are not backed out
    - They are passed to a new UOW
    - If the new UOW abends for any reason the message will be re-queued
  - MQ output messages marked with SYNCPOINT are discarded
  - MQ NO_SYNCPOINT output messages have already been sent

Using the MQ API with IMS Applications

- **Getting the default queue manager name is not straightforward...**
  - MQCONN using default name (blank)
  - MQOPEN the Queue Manager
    - MQOD Objectype = MQOD_Q_MGR
    - MQOD Objectname = blanks
    - MQOO_INQUIRE
  - MQINQ for object name

Using the MQ API with IMS Applications

- STROBE shows MQ CPU in detail by Module/Section
  – Note the expense of MQCONN

```
#PUP                         ** PROGRAM USAGE BY PROCEDURE **

 .SYSTEM      SYSTEM SERVICES              .MQSRIES     MVS/ESA MQSERIES

  MODULE   SECTION    FUNCTION                    % CPU TIME  MARGIN OF ERROR   6.86%
   NAME     NAME                                  SOLO   TOTAL   00       7.00    14.00

  CSQILPLM             MQ DATA MGR SERVICE RTN     .98     .98  **
  CSQLLPLM             MQ LOCK MGR SERVICE RTN    1.47    1.47  ***
  CSQMLPLM             MQ MSG MGR SERVICE RTN     1.47    1.47  ***
  CSQPLPLM             MQ BUFFR MGR SERVICE RT     .49     .49  *
  CSQQCONN CSQQCONN    MQSERIES IMS ADAPTER      12.25   12.25  ******************
  CSQQDISC             MQSERIES IMS ADAPTER       1.96    1.96  ***
  CSQQNORM             MQSERIES IMS ADAPTER        .49     .49  *
  CSQSLD1              MQ STG MGR GLBL MOD EP      .49     .49  *
  CSQWVCOL             MQ IFC RECORD COLLECTIO    1.47    1.47  ***
                                                 -----   -----
   SECTION    .MQSRIES TOTALS:                   21.07   21.07
```

Using the MQ API with IMS Applications

- **In a message driven environment MQ forces a Close/Disconnect and Connect for each message – not each schedule**
  - That is because MQCONN authority is by Userid and each message can be from a different user
  - MQCONN and MQDISC are very expensive and do a lot of I/O to STEPLIB
  - Preloading all of the CSQQxxxx modules in the MQ authorized library eliminated the overhead and STEPLIB access
    - This is an absolute MUST if your MPP transactions issue MQ API calls
    - It is also required for message-driven BMPs
  - A customer reported that preloading CSQACLST, CSQAMLST, and CSQAVICM to do data conversion was helpful

Using the MQ API with IMS Applications

- **In a message driven environment MQ forces a Close/Disconnect and Connect for each message – not each schedule**
  - This can cause problems in a WFI/PWFI environment with Triggered Queues
  - If there are no more messages on the IMS queue and the IMS application does a GU to the IOPCB IMS does not notify MQ for TERM THREAD until the next message arrives or a QC is returned to the IMS application
  - During that time the MQ Queue may still be open
    - MQ internally closes all open queues when it receives TERM THREAD
  - If there are triggered FIRST queues open new messages arriving in MQ will not generate trigger messages because the queue is open
  - To avoid this problem the IMS application should explicitly MQCLOSE any triggered FIRST queues before issuing the next GU to the IOPCB

Using the MQ API with IMS Applications

- **There have been reports of IMS application programs ABENDing with 0C1 when issuing MQ API calls**
  - The main program is an IMS program (ENTRY DLITCBL)
  - It dynamically calls a sub-program which ONLY issues MQ API calls
    - There were no IMS calls
  - The sub-program was NOT linked with the IMS language interface DFSLI000
  - This resulted the ABEND0C1
  - The sub-program must also be linked with DFSLI000 because the MQ API calls are going through the IMS ESS interface

Using the MQ API with IMS Applications

- **In a message driven environment MQ forces a Disconnect and Connect for each message – not each schedule**
  - There is an alternative if your application does not require syncpoint coordination for MQ calls and IMS
    - You can link the application with the MQ batch stub – CSQBSTUB
    - Then a Wait-for-input program can Connect once in the beginning and Disconnect once at the end (but remember previous foil)
    - It can Open queues once in the beginning and Close them once at the end
    - It can issue MQGETs and MQPUTs during IMS transactions
    - It will have to issue MQCMIT calls for any work done “In Syncpoint” from an MQ perspective
  - The first MQCONN in an address space will determine which interface will be used so CSQQSTUB and CSQBSTUB transactions must run in different IMS Message Regions
    - This MPR must also have an SSM member excluding MQ
    - CSQBDEFV can be used to define a default Queue Manager

Using the MQ API with IMS Applications

- **There are several ways the MQ API can be used to have IMS programs interact with MQ queues**
  - MQ IMS Trigger Monitor
  - Customer MQ IMS Trigger Monitor
  - Customer MQ IMS Queue Monitor
  - Enhanced MQ IMS Queue Monitor
  - Customer MQ IMS Queue Processor

MQ IMS Trigger Monitor

- **The MQ IMS Trigger Monitor is an IBM supplied non-Message Driven BMP job which reads "trigger" messages from an MQ Initiation Queue and inserts them to the IMS Message Queue**
  - The IMS application retrieves the trigger message with a GU to the IOPCB
  - The trigger message contains the Queue Manager and Queue Name where the real message resides
  - The IMS application then uses the MQAPI to retrieve the real message
  - The reply message would be done via MQPUT or ISRT to an ALTPCB
    - The reply can not be made to the IOPCB because the message came from a non-message driven BMP

MQ IMS Trigger Monitor

- **These are the steps for the MQ IMS Trigger Monitor**
  1. The MQ IMS Trigger Monitor BMP (CSQQTRMN) is started
  2. MQCONN to the MQ Queue Manager
  3. MQOPEN the Initiation Queue
  4. MQGET with Wait on the Initiation Queue
  5. An MQ application MQPUT’s a message to the triggered queue
  6. MQ generates a trigger message and puts it on the initiation queue
     - Maybe
  7. MQ IMS Trigger Monitor BMP receives the trigger message

MQ IMS Trigger Monitor

- **These are the steps for the MQ IMS Trigger Monitor (continued)**

8. The MQ IMS Trigger Monitor BMP does CHNG/ISRT/PURG of the trigger message to the IMS Queue
9. The MQ IMS Trigger Monitor BMP issues a SYNC call
10. IMS logs the trigger message
11. IMS puts the trigger message in the IMS Message Queue
12. IMS enqueues the trigger message to the IMS transaction
13. The IMS transaction is scheduled in an MPR
14. The IMS transaction does GU to the IOPCB and retrieves the trigger message

MQ IMS Trigger Monitor

- **These are the steps for the MQ IMS Trigger Monitor (continued)**

15. The IMS Transaction does MQCONN for the Queue Manager
16. The IMS Transaction does MQOPEN for the Input Queue
17. The IMS Transaction does MQGET for the real MQ message
18. The IMS Transaction processes the message including IMS and ESAF calls
19. The IMS Transaction does MQPUT1 for the MQ Reply message
20. The IMS Transaction does MQCLOSE for the MQ Input Queue
21. The IMS Transaction does MQDISC to the Queue Manager
22. The IMS Transaction does GU to the IOPCB to create an IMS (and MQ if CSQQSTUB) syncpoint

MQ IMS Trigger Monitor

- **The MQ IMS Trigger Monitor reads the MQ Trigger Message with NO_SYNCPOINT**
  - The Trigger Message is deleted immediately
  - If the BMP ABENDs before its SYNC call or IMS ABENDs before the message gets to the IMS message queue the Trigger Message is gone but the real message is still on the MQ queue
    - If the triggering option was FIRST and this was the last message on the queue there will be no more Trigger Messages and the real message will not be retrieved until the TriggerInterval is reached
    - If the triggering option is EVERY there will not be another trigger message until the next message arrives on the real queue
    - The real message will not be processed until a new trigger message wakes up the MQ IMS Trigger Monitor
  - You could change the first ALTPCB in the CSQQTRMN PSB to EXPRESS so the trigger message will always be sent to IMS

MQ IMS Trigger Monitor

- **IMS application coding consideration**
  - The IMS application must only process ONE real MQ message per GU to the IOPCB to retrieve a trigger message (maybe)
  - Consider this flow
    - GU IOPCB and get trigger message
    - MQCONN
    - MQOPEN
    - MQGET real message
    - Process including IMS and DB2 updates
    - MQPUT1 the reply
    - Go To MQGET until no more messages
  - What could go wrong???
  - What are other options???

MQ IMS Trigger Monitor

- **IMS application coding consideration**
  - What could go wrong?
    - There were no IMS syncpoints in this loop
      - MQCMIT is ignored if using the CSQQSTUB
      - MQCMIT will not commit IMS or DB2 resources
      - You can not issue an IMS CHKP or SYNC call in an MPP
    - If there is an ABEND multiple MQ messages worth of updates may be backed out
      - If MQGET in SYNCPOINT all of the MQ messages are re-queued
      - If MQGET NO SYNCPOINT they have all been freed
    - While you are looping processing the messages all of the IMS and DB2 locks for all of the messages processed are still being held and all of the database buffers are still in use
    - If triggering was EVERY there are trigger messages for which there are no real messages
      - This will result in “false schedules” of IMS transactions

MQ IMS Trigger Monitor

- **What about triggering**
  - If triggering is FIRST and the IMS transaction is processing the real queue and more real messages arrive there will be no more trigger messages
    - But when the real queue is closed – explicitly or implicitly – and there are messages on the real queue then a trigger message will be generated
  - If triggering is EVERY there will be a trigger message for every real message even if the IMS application has the queue open
  - In a Shared MQ Queue environment you may have MQ IMS Trigger Monitors on multiple MQ Queue Managers each waiting on the same Shared Initiation Queue
    - MQ will generate a Trigger Message for EACH MQ Queue Manager that has an MQ IMS Trigger Monitor waiting
      - One IMS application will get the real message
      - One IMS application will have a “false schedule”
      - Please read this:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/topic/com.ibm.mq.csqzal.doc/fg15400_.htm#fg15400___fg15400_1

MQ IMS Trigger Monitor

- **One customer developed a solution to these problems and has given me permission to share it**
  - **1.** GU IOPCB to retrieve a trigger message
    - A. If blank status code go to **2.**
      - IMS will create a syncpoint for the previous unit of work and start a new unit of work
    - B. If QC status code and not no-more-MQ-messages go to **2.**
      - There are no more trigger messages but there may be more MQ messages
      - IMS will create a syncpoint for the previous unit of work and create a new unit of work even though QC was returned
    - C. If QC status code and no-more-MQ-messages then return

MQ IMS Trigger Monitor

- **One customer developed a solution to these problems and has given me permission to share it**
  - **2.** MQCONN
  - **3.** MQOPEN
  - **4.** MQGET
    - If 2033 return code (no message) set no-more-MQ-messages flag and go to **1.**
  - **5.** Process MQ message
  - **6.** MQCLOSE
  - **7.** MQDISC
  - **8.** Go to **1.**
    - Even if there was a previous QC this will work

MQ IMS Trigger Monitor

- **Advantages**
  - It is provided by IBM
  - Only the small trigger message is logged in IMS
  - Only the small trigger message is in the IMS message queue
  - One customer reported that 90% of their 2.8 million transactions per day come in through their 4 MQ IMS Trigger Monitors
- **Disadvantages**
  - **All messages are processed with the Userid of the MQ Trigger Monitor BMP**
  - A Trigger Monitor BMP can only wait on one Initiation Queue
    - But one Initiation Queue can be used for multiple Real queues
  - There are many steps for each message
  - MQ Triggering
    - There are many considerations

Customer IMS Trigger Monitor

- **It is possible to write a Customer IMS Trigger Monitor**
  - This monitor could be written in assembler and wait on multiple Initiation Queues at the same time
  - The one advantage is that it can wait on multiple queues
  - It has all the disadvantages of the IBM MQ IMS Trigger Monitor
  - It also has the disadvantage of being very difficult to write
    - I did write one and it took over a year to program for all of the idiosyncrasies of waiting on multiple ECBs
  - I mention it here so that you will not do it

Customer IMS Queue Monitor

- **It is possible to write a Customer IMS Queue Monitor which reads "real" messages from an MQ Queue and inserts them to the IMS Message Queue**
  - The IMS application retrieves the real message with a GU to the IOPCB
  - The reply message would be done via MQPUT or ISRT to an ALTPCB
    - The reply can not be made to the IOPCB because the message came from a non-message driven BMP

Customer IMS Queue Monitor

- **These are the steps for the Customer IMS Queue Monitor**

1. The Customer IMS Queue Monitor BMP is started
2. MQCONN to the MQ Queue Manager
3. MQOPEN the Real Queue
4. MQGET with Wait on the Real Queue
   - The wait time is short enough to avoid ABENDS522
5. An MQ application MQPUT's a message to the Real Queue
6. Customer IMS Queue Monitor BMP receives the Real message

Customer IMS Queue Monitor

- **These are the steps for the Customer IMS Queue Monitor (continued)**

7. The Customer IMS Queue Monitor BMP does CHNG/ISRT/PURG of the Real message to the IMS Queue
   - May be a multi-segment message
8. The Customer IMS Queue Monitor BMP issues a SYNC call
9. IMS logs the Real message
10. IMS puts the Real message in the IMS Message Queue
11. IMS enqueues the Real message to the IMS transaction
12. The IMS transaction is scheduled in an MPR
13. The IMS transaction does GU to the IOPCB and retrieves the Real message

Customer IMS Queue Monitor

- **These are the steps for the Customer IMS Queue Monitor (continued)**

14. The IMS Transaction processes the message including IMS and ESAF calls
15. The IMS Transaction does MQCONN for the reply message Queue Manager
16. The IMS Transaction does MQPUT1 for the MQ Reply message
17. The IMS Transaction does MQDISC
18. The IMS Transaction does GU to the IOPCB to create an IMS syncpoint

Customer IMS Queue Monitor

- **The Customer IMS Queue Monitor can read the MQ Real Message In SYNCPOINT**
  - The Real Message is not deleted until the IMS SYNC call
  - If the BMP ABENDs before its SYNC call or IMS ABENDs before the message gets to the IMS message queue the MQ message is re-queued
    - The number of times this happens will be shown in MQMD_BackOutCount

Customer IMS Queue Monitor

- **The Customer IMS Queue Monitor may have to pass the Reply-to Queue and Reply-to Queue Manager information to the IMS transaction**
  - The IMS application does not do the MQGET for the real message and does not get the MQMD
  - The Customer IMS Queue Monitor can insert an extra IMS message segment
    - Could pass just the Reply-to information
    - Could pass the entire MQMD

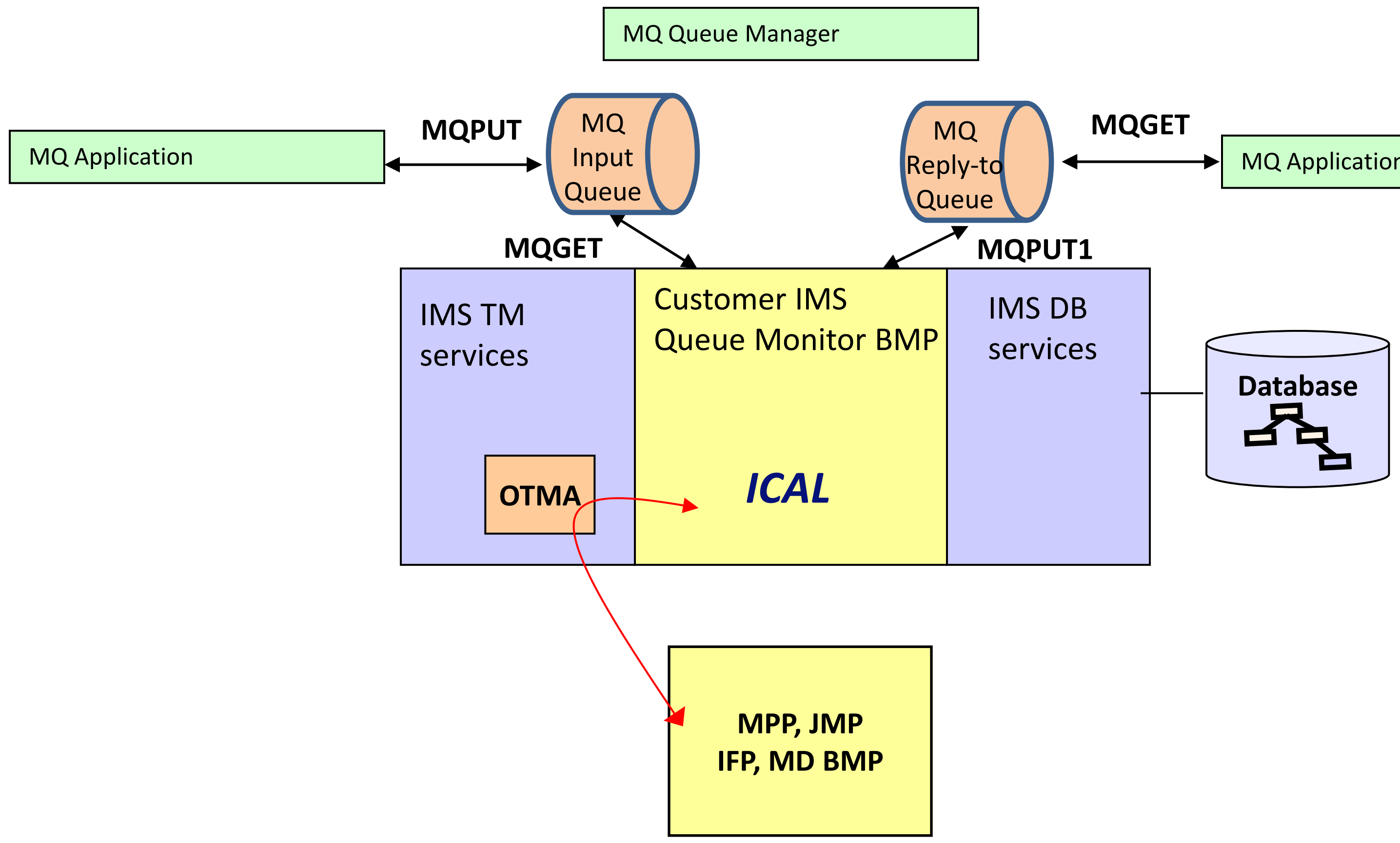
Customer IMS Queue Monitor

- **Advantages**
  - Less overhead in the IMS MPR than a trigger monitor
  - No MQ Triggering complications and overhead
- **Disadvantages**
  - All messages are processed with the Userid of the Customer IMS Queue Monitor
  - The Customer IMS Queue Monitor can only wait on one Real Queue
    - But there can be multiple BMP’s reading the same queue
  - Someone has to add the llzzTRANCODE
    - MQ application
    - Customer IMS Queue Monitor
  - The Real MQ message may have to be segmented
    - If > 32K
  - The Real MQ message is logged in IMS
    - This could be VERY large
  - The Real MQ message goes in the IMS message queue
    - This could be VERY large

Enhanced IMS Queue Monitor

- **It is possible to write an Enhanced IMS Queue Monitor which reads “real” messages from an MQ Queue and sends them to the IMS transaction using IMS 13 Synchronous Program Switch**
  - The IMS application retrieves the real message with a GU to the IOPCB
  - The IMS application inserts the reply to the IOPCB
  - The BMP receives the reply message
  - The BMP sends the reply message via MQPUT1 to the Reply-to Queue and Queue Manager

Enhanced IMS Queue Monitor



Enhanced IMS Queue Monitor

- **These are the steps for the Enhanced IMS Queue Monitor**

1. The Enhanced IMS Queue Monitor BMP is started
2. MQCONN to the MQ Queue Manager
3. MQOPEN the Real Queue
4. MQGET with Wait on the Real Queue
    - The wait time is short enough to avoid ABENDS522
5. An MQ application MQPUT’s a message to the Real Queue
6. The Enhanced IMS Queue Monitor BMP receives the Real message

Enhanced IMS Queue Monitor

- **These are the steps for the Enhanced IMS Queue Monitor (continued)**

7. The Enhanced IMS Queue Monitor sends the message to the IMS application via Synchronous Program Switch
8. IMS logs the Real message
9. IMS puts the Real message in the IMS Message Queue
10. IMS enqueues the Real message to the IMS transaction
11. The IMS transaction is scheduled in an MPR
12. The IMS transaction does GU to the IOPCB and retrieves the Real message

Enhanced IMS Queue Monitor

- **These are the steps for the Enhanced IMS Queue Monitor (continued)**

13. The IMS Transaction processes the message including IMS and ESAF calls
14. The IMS Transaction inserts the reply message to the IOPCB
15. IMS logs the Reply message
16. IMS puts the Reply message in the IMS Message Queue
17. The IMS Transaction does GU to the IOPCB to create an IMS syncpoint
18. The Enhanced IMS Queue Monitor does MQPUT1 for the MQ Reply message
19. The Enhanced IMS Queue Monitor issues a SYNC call

Enhanced IMS Queue Monitor

- **The Customer IMS Queue Monitor can read the MQ Real Message In SYNCPOINT**
  - The Real Message is not deleted until the IMS SYNC call
  - If IMS or the BMP ABEND before its SYNC call the MQ message is re-queued
    - The number of times this happens will be shown in MQMD_BackOutCount

Enhanced IMS Queue Monitor

- **The Enhanced IMS Queue Monitor does not have to pass the Reply-to Queue and Reply-to Queue Manager information to the IMS transaction**
  - The Enhanced IMS Queue Monitor has the MQMD
  - The Enhanced IMS Queue Monitor can send the reply to the Reply-to Queue and Queue Manager

Enhanced IMS Queue Monitor

- **Advantages**
  - Less overhead in the IMS MPR than a trigger monitor
  - No MQ Triggering complications and overhead
  - No need to set up the MQ OTMA interface (MQ IMS Bridge)
  - No changes to existing IMS applications
- **Disadvantages**
  - All messages are processed with the Userid of the Customer IMS Queue Monitor
  - The Customer IMS Queue Monitor can only wait on one Real Queue
    - But there can be multiple BMP's reading the same queue
  - Someone has to add the llzzTRANCODE
    - MQ application
    - Enhanced IMS Queue Monitor
  - The Real MQ input and reply messages are logged in IMS
    - These could be VERY large
  - The Real MQ input and reply messages go in the IMS message queue
    - These could be VERY large

Customer IMS Queue Processor

- **It is possible to write a Customer IMS Queue Processor which reads “real” messages from an MQ Queue and does all of the processing within the BMP itself**
  - There is no message switching to an IMS transaction
  - The reply message would be done via MQPUT or ISRT to an ALTPCB
  - This is the most efficient way for IMS applications to process MQ messages using the MQ API

Customer IMS Queue Processor

- **These are the steps for the Customer IMS Queue Processor**

  1. The Customer IMS Queue Processor BMP is started
  2. MQCONN to the MQ Queue Manager
  3. MQOPEN the Real Queue
  4. MQGET with Wait on the Real Queue
     - The wait time is short enough to avoid ABENDS522
  5. An MQ application MQPUT’s a message to the Real Queue
  6. Customer IMS Queue Processor BMP receives the Real message

Customer IMS Queue Processor

- **These are the steps for the Customer IMS Queue Processor (continued)**

7. The Customer IMS Queue Processor processes the message including IMS and ESAF calls
   - It may have to call different subroutines for different “transaction codes”
8. The Customer IMS Queue Processor does MQPUT1 for the MQ Reply message
9. The Customer IMS Queue Processor does an IMS SYNC call
10. The Customer IMS Queue Processor loops to do another MQGET with Wait

Customer IMS Queue Processor

- **The Customer IMS Queue Processor can read the MQ Real Message In SYNCPOINT**
  - The Real Message is not deleted until the IMS SYNC call
  - If the BMP ABENDs before its SYNC call or IMS ABENDs before the message gets to the IMS message queue the MQ message is re-queued
    - The number of times this happens will be shown in MQMD_BackOutCount

Customer IMS Queue Processor

- **The Customer IMS Queue Processor does not have to pass the Reply-to Queue and Reply-to Queue Manager information to the IMS transaction**
  - The input MQMD is available

Customer IMS Queue Processor

- **Advantages**
  - No IMS MPR overhead
  - No IMS logging of the MQ messages
  - No IMS message on the IMS Queue
  - No MQ Triggering complications and overhead
  - No llzzTRANCODE
  - No message segmentation

Customer IMS Queue Processor

- **Disadvantages**
  - All processing is done with the Userid of the Customer Queue Processor
  - Volume may require more than one Customer IMS Queue Processor
    - This is not a problem
      - There can be multiple Customer Queue Processor BMPs reading the same queue
      - No different than multiple MPPs processing these messages
  - The Customer IMS Queue Processor can only wait on one Real Queue
    - But there can be multiple BMP’s reading different queues

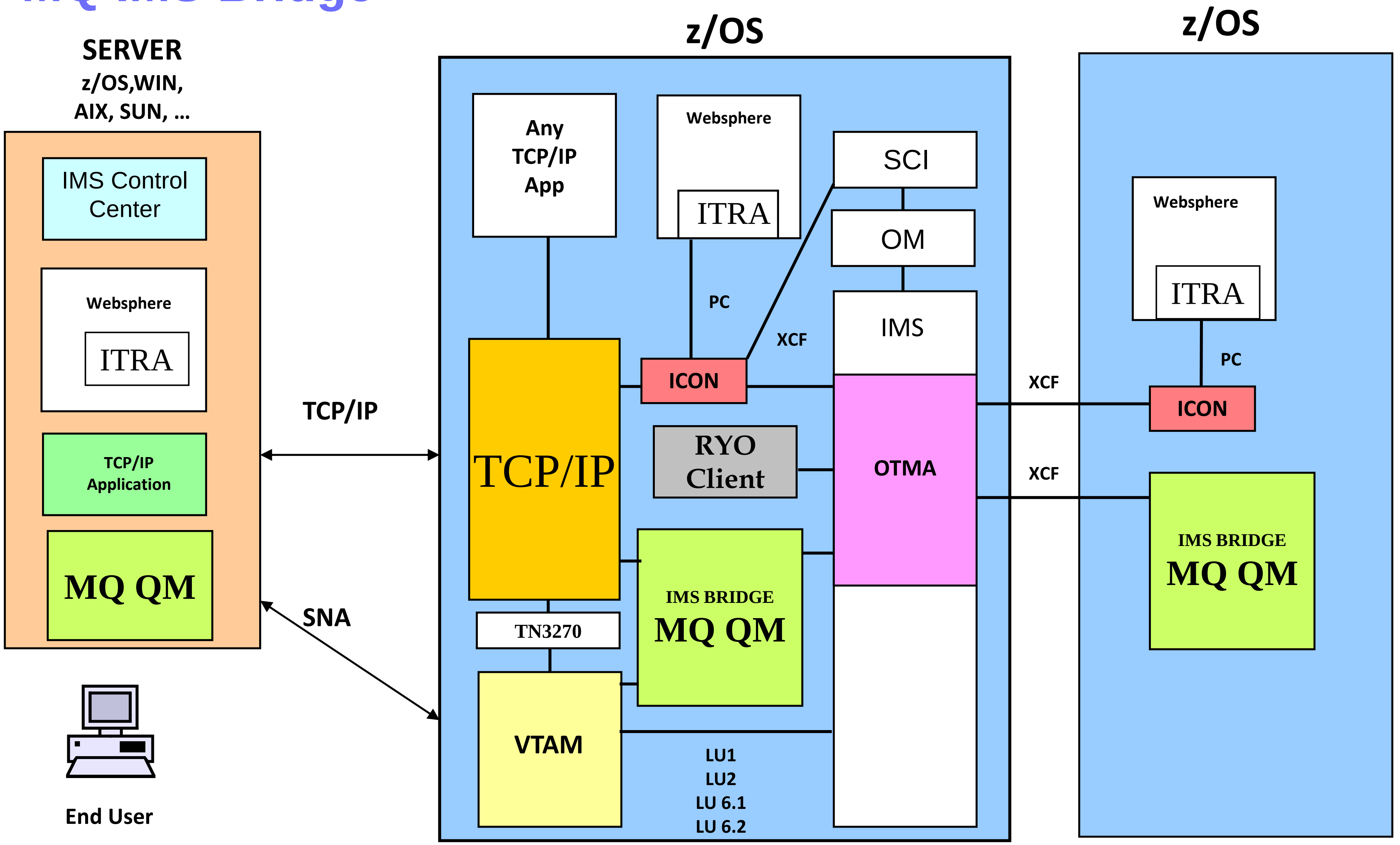
Alternatives for Using the MQ API Summary

- **There are several ways the MQ API can be used to have IMS programs interact with MQ queues**
  - MQ IMS Trigger Monitor
  - Customer MQ IMS Trigger Monitor
  - Customer MQ IMS Queue Monitor
  - Customer MQ IMS Queue Processor
    - This is my favorite

MQ IMS Bridge

- **This is code in the MQ Queue Manager**
  - The IMS Bridge is an OTMA client
    - For specially defined queues it will MQGET the messages from the queue and send them to IMS using the IMS OTMA interface
  - The IMS bridge also gets output messages from IMS via the OTMA interface
    - IOPCB output
      - The output message is MQPUT to the Reply-to Queue and Reply-to Queue Manager in the original MQ input message MQMD passed and returned in the OTMA Prefix User Data
    - ALTPCB output
      - The output message is MQPUT to the Reply-to Queue and Reply-to Queue Manager in the MQMD created by the OTMA DRU exit and returned in the OTMA Prefix User Data

MQ IMS Bridge



MQ IMS Bridge

- **MQ IMS Bridge**
  - One MQ queue manager can connect to multiple IMS control regions
  - One IMS control region can connect to multiple MQ queue managers
  - MQ and all of the IMS Control Regions it connects to must be in the same XCF group
  - MQ and IMS can be on different LPARs in the same Sysplex
  - MQ IMS Bridge start and stop events are sent to the SYSTEM.ADMIN.CHANNEL.EVENT.QUEUE

MQ IMS Bridge

- **Define MQ to OTMA in CSQZPARM**
  - OTMACON keyword on CSQ6SYSP macro
    - OTMACON(Group,Member,Druexit,Age,TPIPEPrefix)
      - Group = XCF group
      - Member = MQ XCF member (OTMA TMEMBER)
      - Druexit = IMS exit to format OTMA User Data (overrides DFSYDTx)
        - Consider a name of DRU0xxxx (xxxx = MQ Queue Manager name)
      - Age = how long a Userid (ACEE) from MQ is valid in the OTMA cache before it expires
      - TPIPEPrefix = three character prefix for TPIPE name
        - To avoid collision with IMS transaction code names
        - Two characters for MQ shared queues
  - Member CSQ4ZPRM in data set hlq.SCSQPROC has default CSQZPARM members you can use to build your members
  - My strong requirement is that all of these should be able to be specified (and used!!!) on the STGCLASS definition

MQ IMS Bridge

- **MQ IMS Bridge**
  - When the message arrives in MQ it will be sent via XCF to the IMS OTMA interface
  - Message may be:
    - an IMS transaction
    - an IMS command (only a subset of commands are allowed)
    - NOT a message to an IMS LTERM
  - IMS will put it on the IMS message queue
  - The application will do a GU to the IOPCB to retrieve the message
    - This is very similar to the implicit LU6.2 process
    - **There are no changes to existing IMS programs**
      - ALTPCB output may have to be routed by OTMA exits or OTMA Descriptors
  - A remote queue manager can send a message to a local queue destined for IMS via OTMA

MQ IMS Bridge

- **These are the steps for the MQ IMS Bridge**

1. An MQ application MQPUT's a message to the IMS Bridge Queue specifying a Reply-to Queue and Queue Manager in the MQMD
2. The MQ IMS Bridge sends the message to OTMA via XCF
   - The MQMD is sent in the OTMA User Data
3. IMS logs the Real message
4. IMS puts the Real message in the IMS Message Queue
5. IMS enqueues the Real message to the IMS transaction
6. The IMS transaction is scheduled in an MPR
7. The IMS transaction does GU to the IOPCB and retrieves the Real message
8. The IMS transaction inserts the reply to the IOPCB

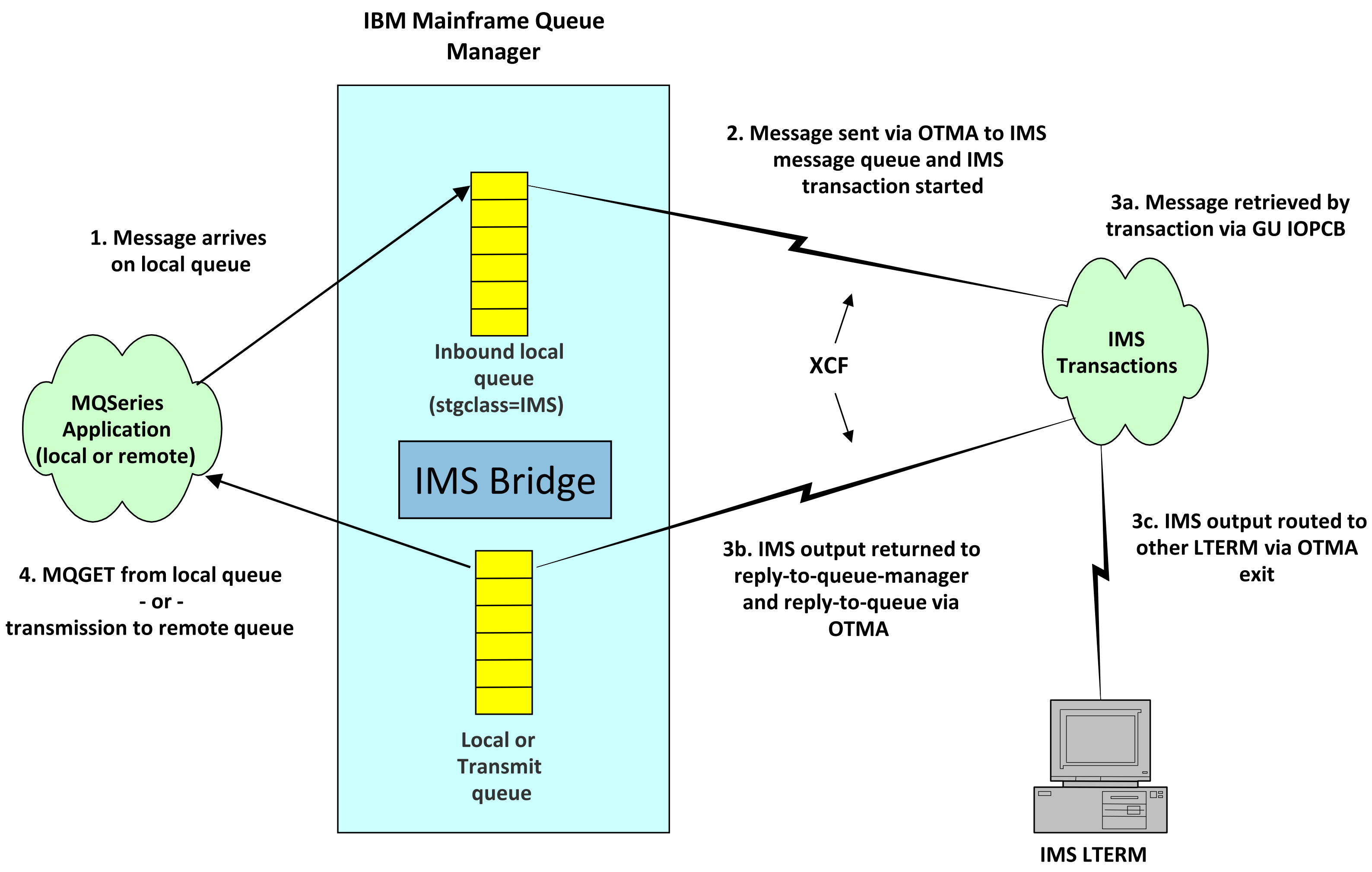
MQ IMS Bridge

- **These are the steps for the MQ IMS Bridge**

9. IMS logs the Reply message
10. IMS puts the Reply message in the IMS Message Queue
11. IMS enqueues the Reply message to the output TPIPE
    - If Commit Mode 0
12. IMS OTMA sends the reply message to MQ via XCF
    - The MQMD is returned in the OTMA User Data
13. The MQ IMS Bridge puts the message on the Reply-to Queue

There is no change to the IMS application

MQ IMS Bridge


IBM Mainframe Queue Manager
1. Message arrives on local queue
MQSeries Application (local or remote)
Inbound local queue (stgclass=IMS)
IMS Bridge
Local or Transmit queue
2. Message sent via OTMA to IMS message queue and IMS transaction started
XCF
3a. Message retrieved by transaction via GU IOPCB
IMS Transactions
3b. IMS output returned to reply-to-queue-manager and reply-to-queue via OTMA
3c. IMS output routed to other LTERM via OTMA exit
IMS LTERM
4. MQGET from local queue - or - transmission to remote queue

MQ IMS Bridge

- **Asynchronous output (ALTPCB output) can also be send to MQ via OTMA**
  - Requires the use of IMS exits or OTMA descriptors (IMS 13)
- **There are many other setup parameters and considerations for using the MQ IMS Bridge**
  - These are topics for another presentation

MQ IMS Bridge

- **Advantages**
  - Less overhead in the IMS MPR than using the MQ API
  - No MQ Triggering complications and overhead
  - All messages are processed with the Userid in the OTMA Header
  - There are no changes to existing IMS applications
- **Disadvantages**
  - The MQ application has to provide the llzzTrancode
  - The MQ application may have to do additional segmenting
  - The Real MQ message is logged in IMS
    - This could be VERY large
  - The Real MQ message goes in the IMS message queue
    - This could be VERY large

Questions?