

# Unit 9 - 1

## More IMS Connectivity

# IMS Application Access to WebSphere MQ Messages

## 1. Connecting WebSphere MQ and IMS via ESS

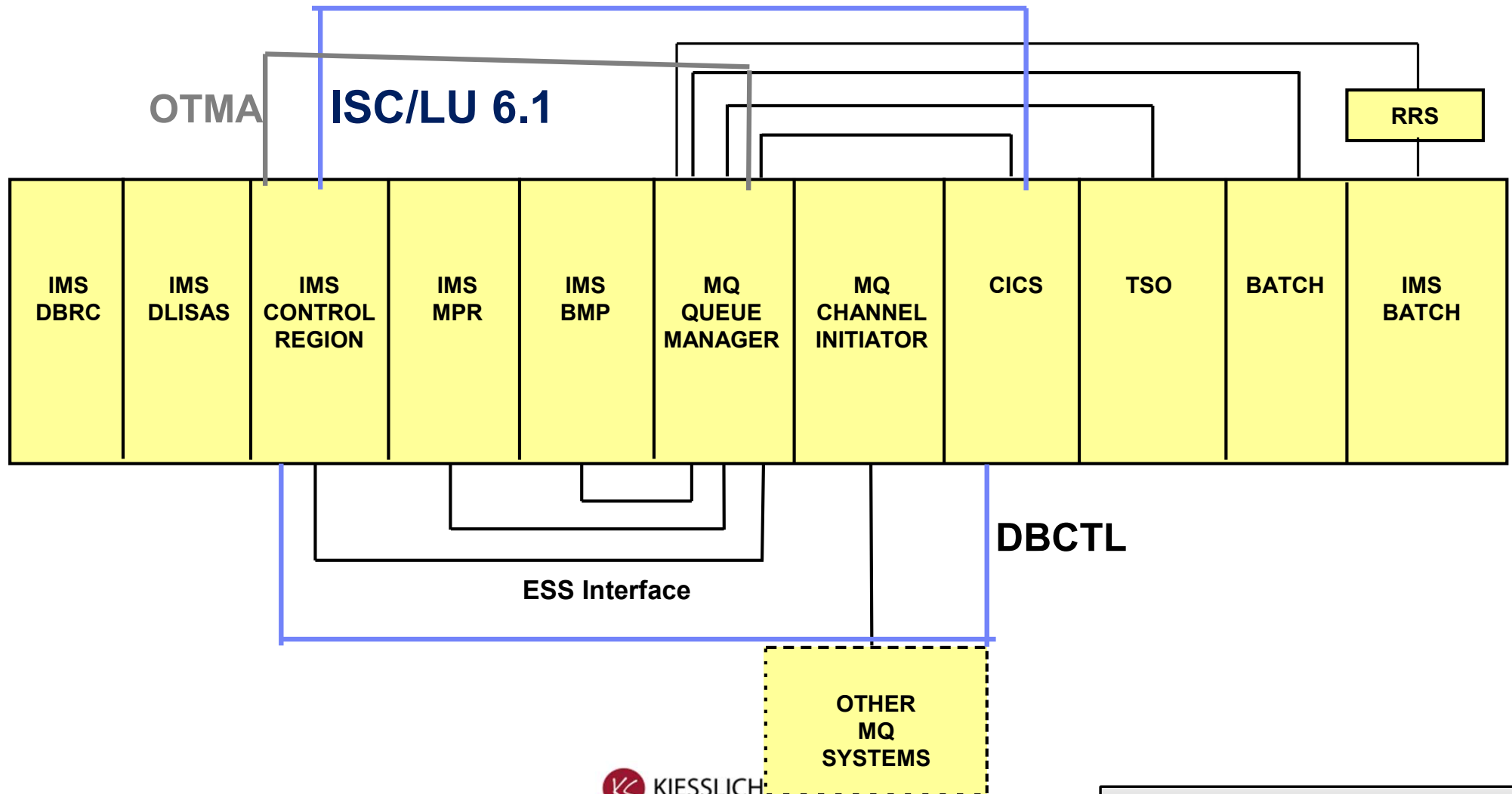
- Using the MQ API with IMS Applications

## 2. WebSphere MQ IMS Bridge

- WebSphere MQ IMS Bridge Security

# Connecting WebSphere MQ and IMS via ESS

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



# Connecting W/MQ and IMS via ESS

Define WebSphere MQ to IMS by adding ESS information to the IMS PROCLIB (member name IMIDxxxx)

**FORMAT: SST=,SSN=,LIT=,ESMT=,RTT=,REO=,CRC=**

**SST: Subsystem Type - “MQS”**

**SSN: Subsystem Name - MQ subsystem**

**LIT: Language Interface Token - See CSQQDEFV**

**ESMT: External Subsystem Module Table - “CSQQESMT”**

**RTT: Resource Translation Table - Not Used by MQ**

**REO: Region Error Option - “R”, “Q”, or “A”**

**CRC: Subsystem Recognition Character - Not Used by MQ**

— The /SSR command is not supported

# Connecting W/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 an authorized library in the IMS control region STEPLIB concatenation

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

# Using the WebSphere MQ API with IMS Applications

## WebSphere 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
  - CSQQSTUB
    - IMS stub
    - WebSphere MQ knows the application is running in an IMS environment
    - Provides two-phase commit for IMS and MQ API calls
  - CSQBSTUB
    - Batch stub
    - WebSphere MQ does not know the application is running in an IMS environment
    - There is no two-phase commit with IMS

# Using the WebSphere MQ API with IMS Applications (2)

**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 WebSphere MQ API with IMS Applications (3)

## IMS and MQ Units of Work

- An IMS commit is also an MQ and DB2 commit
- SYNC, CHKP, GU to IOPCB (MODE=SNGL), normal pgm 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
  - 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
    - 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 for security reasons ( Connection security is by Userid , Each message can be from a different Userid )*



# Using the WebSphere MQ API with IMS Applications (4)

## 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      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 WebSphere MQ API with IMS Applications (5)

**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
- Another customer reported that preloading CSQACLST, CSQAMLST, and CSQAVICM to do data conversion was helpful

# Using the WebSphere MQ API with IMS Applications (6)

**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 WebSphere MQ API with IMS Applications (7)

**There are several ways the MQ API can be used to have IMS programs interact with MQ queues**

- WebSphere MQ IMS Trigger Monitor (classic)
- Customer MQ IMS Trigger Monitor
- Customer MQ IMS Queue Monitor
- Customer MQ IMS Queue Processor

# WebSphere MQ IMS Trigger Monitor (1)

**The W/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**

# WebSphere MQ IMS Trigger Monitor (2)

These are the steps for the MQ IMS Trigger Monitor

- The MQ IMS Trigger Monitor BMP (CSQQTRMN) is started
- MQCONN to the MQ Queue Manager
- MQOPEN the Initiation Queue
- MQGET with Wait on the Initiation Queue
- An MQ application MQPUT's a message to the triggered queue
- MQ generates a trigger message and puts it on the initiation queue
- MQ IMS Trigger Monitor BMP receives the trigger message

# WebSphere MQ IMS Trigger Monitor (3)

... (continued)

- The MQ IMS Trigger Monitor BMP does CHNG/ISRT/PURG of the trigger message to the IMS Queue
- The MQ IMS Trigger Monitor BMP issues a SYNC call
- IMS logs the trigger message
- IMS enqueues the trigger message to the IMS transaction
- The IMS transaction is scheduled in an MPR
- The IMS transaction kicks off the application doing GU to the IOPCB and retrieves the trigger message
- The IMS Transaction does MQCONN for the Queue Manager
- The IMS Transaction does MQOPEN for the Input Queue

# WebSphere MQ IMS Trigger Monitor (4)

... (continued)

- The IMS Transaction does MQGET for the real MQ message
- The IMS Transaction processes the message including IMS and ESS calls (f.i. DB2)
- The IMS Transaction does MQPUT1 for the MQ Reply message
- The IMS Transaction does MQCLOSE for the MQ Input Queue
- The IMS Transaction does MQDISC to the Queue Manager
- The IMS Transaction does GU to the IOPCB to create an IMS syncpoint



# WebSphere MQ IMS Trigger Monitor (5)

- **Advantages**

- ✓ It is provided by IBM
- ✓ Only the small trigger message is logged in IMS
- ✓ One customer reported that 90% of their 2.8 millions transactions per day come in through their 4 MQ IMS Trigger Monitors

- **Disadvantages**

- A Trigger Monitor BMP can only wait on one Initiation Queue
- There are many steps for each message
- WebSphere MQ Triggering
  - There are many, 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

# Customer IMS Queue Monitor (1)

**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 (2)

## These are the steps for the Customer IMS Queue Monitor

- The Customer IMS Queue Monitor BMP is started
- MQCONN to the MQ Queue Manager
- MQOPEN the Real Queue
- MQGET with Wait on the Real Queue
- An MQ application MQPUT's a message to the Real Queue
- Customer IMS Queue Monitor BMP receives the Real message

# Customer IMS Queue Monitor (3)

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

- The Customer IMS Queue Monitor BMP does CHNG/ISRT/PURG of the Real message to the IMS Queue
- The Customer IMS Queue Monitor BMP issues a SYNC call
- IMS logs the Real message
- IMS enqueues the Real message to the IMS transaction
- The IMS transaction is scheduled in an MPR
- The IMS transaction does GU to the IOPCB and retrieves the Real message

# Customer IMS Queue Monitor (4)

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

- The IMS Transaction processes the message including IMS and ESAF calls
- The IMS Transaction does MQCONN for the reply message Queue Manager
- The IMS Transaction does MQPUT1 for the MQ Reply message
- The IMS Transaction does MQDISC
- The IMS Transaction does GU to the IOPCB to create an IMS syncpoint

# Customer IMS Queue Monitor (5)

## 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 (6)

**The Customer IMS Queue Monitor may have to pass the Reply-to Queue and Reply-to Queue Manager information to the IMS transaction**

- This can be done by inserting an extra IMS message segment
  - Could pass just the Reply-to information
  - Could pass the entire MQMD



# Customer IMS Queue Monitor (7)

## Advantages

- ✓ Less overhead in the IMS MPR
- ✓ No MQ Triggering complications and overhead

## Disadvantages

- The Customer IMS Queue Monitor can only wait on one Real Queue
  - But there can be multiple BMP's reading the same queue
- The Real MQ message is logged in IMS
  - This could be VERY large
- The Real MQ message goes on the IMS message queue
  - This could be VERY large

# Customer IMS Queue Processor (1)

**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 (2)

## These are the steps for the Customer IMS Queue Processor

- The Customer IMS Queue Processor BMP is started
- MQCONN to the MQ Queue Manager
- MQOPEN the Real Queue
- MQGET with Wait on the Real Queue
- An MQ application MQPUT's a message to the Real Queue
- Customer IMS Queue Processor BMP receives the Real message

# Customer IMS Queue Processor (3)

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

- The Customer IMS Queue Processor processes the message including IMS and ESAF calls
- The Customer IMS Queue Processor does MQPUT1 for the MQ Reply message
- The Customer IMS Queue Processor does an IMS SYNC call
- The Customer IMS Queue Processor loops to do another MQGET with Wait

# Customer IMS Queue Processor (4)

## 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 (5)

## 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

## Disadvantages

- The Customer IMS Queue Processor can only wait on one Real Queue
  - But there can be multiple BMP's reading the same queue

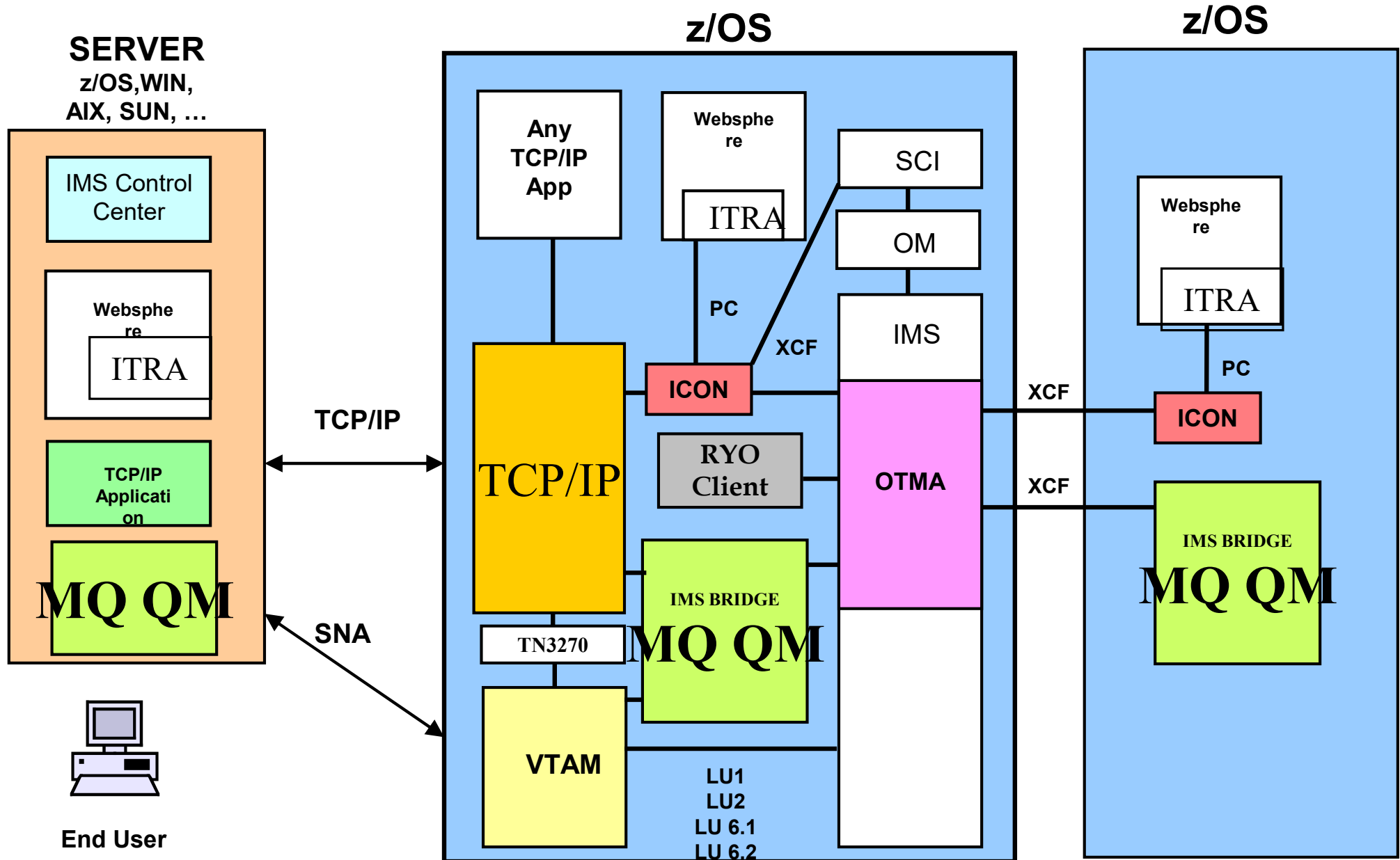
# WebSphere MQ IMS Bridge (OTMA only)

(the easiest way)

## 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

# WebSphere MQ IMS Bridge (OTMA only) - 2





# WebSphere MQ IMS Bridge (OTMA only) - 3

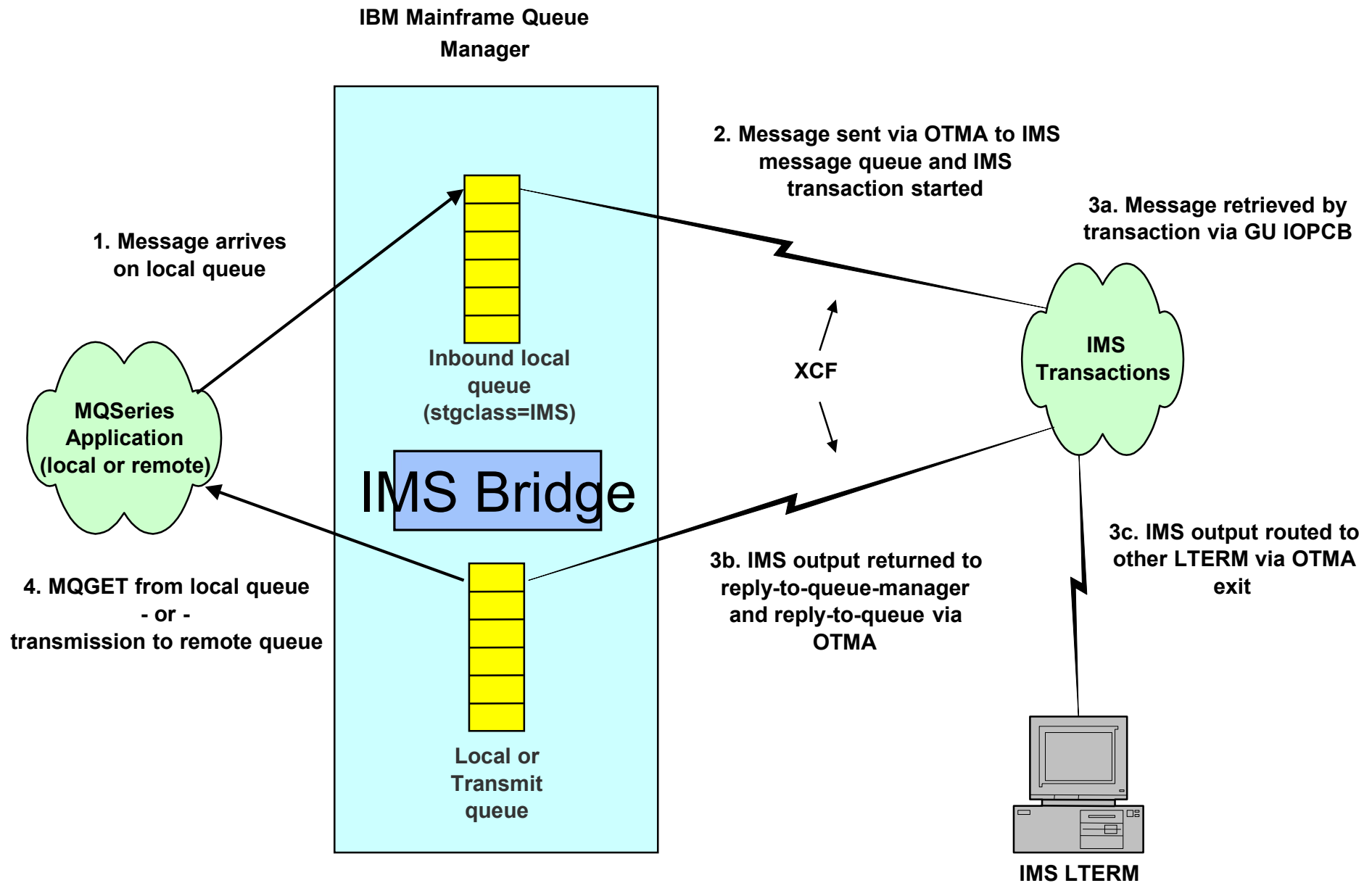
## WebSphere MQ IMS Bridge

- 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 (XCF)
  - WebSphere MQ IMS Bridge start and stop events are sent to the `SYSTEM.ADMIN.CHANNEL.EVENT.QUEUE`

# WebSphere MQ IMS Bridge (OTMA only) - 4

- 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

# WebSphere MQ IMS Bridge (OTMA only) - 5



# WebSphere MQ IMS Bridge (OTMA only) - 6

**It is possible to build a “synchronous” MQ application accessing an IMS transaction**

- Issue MQPUT to the IMS Bridge Queue
- Issue MQCMIT to commit the message
  - Or MQPUT not in syncpoint
- Issue MQGET with wait on the reply-to queue
- The IMS Bridge sends the message to IMS via OTMA
- The transaction is processed and responds
- IMS sends the reply to the IMS Bridge via OTMA
- The IMS Bridge puts the response on the reply-to queue
- The MQGET is now completed
- Issue MQCMIT to commit the reply message

# WebSphere MQ IMS Bridge (OTMA only) - 7

## 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

# WebSphere MQ IMS Bridge (OTMA only) - 8

## Define MQ to OTMA in CSQZPARM

### — Druexit

- If null is specified then MQ will pass the name “DFSYDRU0”
  - OTMACON(GROUP1,MQTMBR,,5000,PFX) (\*)
- If MQ does not need this exit there is a way to avoid the error message
  - Code the Druexit as one blank in single quotes
    - > OTMACON(GROUP1,MQTMBR,' ',5000,PFX)

(\*)

After IMS APAR PK25454 (UK17882) IMS will issue message DFS1269E SEVERE IMS INTERNAL FAILURE, REASON CODE=3432 if it can not load a DRU exit specified during a client bid. Before this if the load failed no message was issued. This was changed to a more user friendly message via PK53291/PK61265.

# WebSphere MQ IMS Bridge (OTMA only) - 9

**Define one or more storage classes with the XCFGNAME and XCFMNAME parameters of the IMS systems to which you will connect**

- DEFINE STGCLASS(IMSA) -
- PSID(02) –
- XCFGNAME(XCFGROUP) -
- XCFNAME(XCFIMSA) –
- PASSTKTA(applname) (6.0)
- The XCFGNAME will not be used
  - The one in the ZPARM will be used
- When a STGCLASS is defined for a new IMS MQ will attempt to establish the connection

# WebSphere MQ IMS Bridge (OTMA only) - 10

## Changing the Storage Class of an IMS Bridge Queue must be done carefully

- If there are CM0 messages on the IMS Bridge Queue in MQ
  - Do not ALTER the Queue to a different Storage Class pointing to a different IMS copy
    - It will cause sequence number errors (in both IMS copies)
  - Have two different IMS Bridge Queues each with a different Storage Class for the different IMS Copies
    - You can define an Alias Queue which you can point to one IMS Bridge Queue or the other so that the MQ application only has to MQPUT to one queue name
- If there are no CM0 messages on the IMS Bridge Queue you can ALTER it to point to a different IMS Storage Class



# WebSphere MQ IMS Bridge (OTMA only) - 11

## Define queues

- Define local queue(s) referencing the storage classes
  - DEFINE QLOCAL(MQID\_TO\_IMSA) –
- STGCLASS(IMSA)
- Define reply-to queue(s)
  - DEFINE QLOCAL(MQID\_FROM\_IMSA)
  - These could also be remote queues

# WebSphere MQ IMS Bridge (OTMA only) - 12

## Operating the WebSphere MQ IMS Bridge

- After startup MQ will join the XCF group defined in the OTMACON parameter
- The IMS Bridge will initiate a client bid resync to each active IMS defined in the STGCLASS macros
- When the bid is successful the IMS Bridge will open the Bridge Queues and messages will flow
- If a new STGCLASS for IMS is added MQ will attempt to connect to the IMS

# WebSphere MQ IMS Bridge (OTMA only) - 13

## Operating the WebSphere MQ IMS Bridge

- If you GET DISABLE an IMS Bridge Queue it will stop messages from MQ to IMS for just that queue
- New commands were added with MQSeries V6.0
  - SUSPEND QMGR BRIDGE(IMS)
    - Stops MQGETs by the IMS Bridge from all IMS Bridge Queues
    - Allows MQPUTs to the reply-to queue by the IMS Bridge of replies from IMS
  - RESUME QMGR BRIDGE(IMS)
    - Starts MQGETs by the IMS Bridge from IMS Bridge Queues

# WebSphere MQ IMS Bridge (OTMA only) - 14

## Operating the WebSphere MQ IMS Bridge ...cont.

- There are IMS commands that affect the WebSphere MQ IMS Bridge
  - /STOP OTMA
    - Closes the connection to all OTMA clients
  - /START OTMA
    - Opens the connection to all OTMA clients
    - Any clients already in the XCF group (e.g. MQ) are notified
  - /STOP TMEMBER xxxx TPIPE yyyy
    - OTMA sends a message to the OTMA client to stop input for the TPIPE
    - OTMA suspends sending output to the TPIPE
  - /START TMEMBER xxxx TPIPE yyyy
    - OTMA sends a message to the OTMA client to start input for the TPIPE
    - OTMA resumes sending output to the TPIPE
  - There are no /STOP TMEMBER or /START TMEMBER commands

# WebSphere MQ IMS Bridge (OTMA only) - 15

**MQ creates two TPIPEs per local queue defined as using the IMS Bridge**

**— *One is for “asynchronous” messages***

- Commit mode 0 - commit-then-send
- Output is “asynchronous” to transaction completion
- This is a SYNChronized TPIPE
  - Messages sent with valid sequence numbers are recoverable after subsystem failures
- The TPIPE name for non-shared MQ queues is xxx0nnnn
  - xxx = User defined prefix
- The TPIPE name if using MQ shared queues is xx0nnnnn
  - xx = User defined prefix

# WebSphere MQ IMS Bridge (OTMA only) - 16

**MQ creates two TPIPEs per local queue defined as using the IMS Bridge**

- *One is for “synchronous” messages*
  - Commit mode 1 - send-then-commit
  - Output is “synchronous” with transaction completion
  - This is a non-SYNChronized TPIPE
  - The TPIPE name for non-shared queues is xxx8nnnn
    - xxx = User defined prefix
  - The TPIPE name for shared queues is xx8nnnnn
    - xx = User defined prefix
  - Required for EMH, Conversational, and IMS commands
- The TPIPEs are created when the first message for the type (sync, async) arrives on the IMS Bridge queue

# WebSphere MQ IMS Bridge (OTMA only) - 17

## DISPLAY QL(name) TPIPE

- MQ has a TPIPE keyword to the DISPLAY QL command
  - Display the TPIPE names for local queues
  - ALL local queues have TPIPE names assigned – not just IMS Bridge Queues
    - their Storage Class may be altered later to IMS

```
!MQ37DIS QL(SYSTEM.DEFAULT.LO*) TPIPE
      CSQM293I !MQ37 CSQMDRTC 1 QLOCAL FOUND MATCHING REQUEST
      CSQM201I !MQ37 CSQMDRTC  DIS QLOCAL DETAILS
      QUEUE(SYSTEM.DEFAULT.LOCAL.QUEUE)
      TYPE(QLOCAL)
      QSGDISP(QMGR)
      TPIPE(
        CSQ00000
        CSQ80000
      )
      END QLOCAL DETAILS
      CSQ9022I !MQ37 CSQMDRTC ' DIS QLOCAL' NORMAL COMPLETION
```

# WebSphere MQ IMS Bridge (OTMA only) - 18

## There is a limit to the capacity of an IMS TPIPE

- There are several factors involved
  - SYNChronized versus non-SYNChronized (CM0 versus CM1)
  - MQ Persistent versus non-Persistent
  - IMS RECOVER versus NONRECOVER
  - Message size
- More capacity will require more TPIPEs which will require more MQ queues
  - There are only 2 TPIPES per queue
  - The application has to round-robin the messages to the queues



# WebSphere MQ IMS Bridge - Input MSGs

**The data stream passed to MQ by the application MQPUT is in LLZZTrancodeDataLLZZData... Format (typical IMS input msg layout)**

- This allows for multi-segment input messages via OTMA to IMS
- The IMS Bridge will create IMS segments for each LLZZdata
- The MQMD.Format of MQFMT\_IMS\_VAR\_STRING (“MQIMSVS”) or MQFMT\_IMS (“MQIMS”) tells MQ that the data contains LLZZ’s

# WebSphere MQ IMS Bridge - Input MSGs (2)

## The sending application can optionally provide an IMS sub-header (MQIIH)

- Specify the presence of the MQIIH sub-header by using the MQMD.Format=MQFMT\_IMS (“MQIMS”)
  - WARNING: MQIIH must be fullword aligned
- An output MQIIH will be returned with the output message
- This header specifies IMS and MQ parameters
  - See the next two foils
- The format of this input message is:
  - MQIIHLLZZTrancodeDataLLZZData...

# WebSphere MQ IMS Bridge - Input MSGs (3)

## MQIIH parameters are:

- Several reserved fields...
- IMS LTERM name to put in the IOPCB
- MFS Format name to put in the IOPCB
  - No MFS formatting is actually done
    - Reply-To Format (MQ format name)
    - Authenticator (RACF password or PassTicket)
    - Transaction Instance ID (if IMS conversational)
    - Transaction State (conversational or not, architected command)
    - Commit Mode (CM0 or CM1)
    - Security Scope (Check or Full) -> Only honored if /SEC OTMA PROFILE used
    - Flags
      - *Pass Expiration*
      - *Reply Format None*

# WebSphere MQ IMS Bridge - Input MSGs (4)

If **no** MQIIH is presented to the IMS Bridge  
(MQMD.Format=MQFMT\_IMS\_VAR\_STRING) then  
default values are assumed:

- LTERM=TPIPEName
- MODNAME=MQMD.Format
  - Default is “MQIMSVS”
- MQMD.Format is used as the MFSTMapName
- Non-conversational
- Commit-then-send (commit mode 0)
- Security mode is MQISS\_CHECK
- All flags are off

# WebSphere MQ IMS Bridge - Input MSGs (5)

**MQ allows the addition of one or more standard MQ User Headers to be passed with messages going to an IMS Bridge Queue**

- The user header(s) will be passed to IMS in the OTMA User Data
- IMS will log this data and pass it back in the OTMA User Data for the reply from the IOPCB
  - ALTPCB output will probably NOT have this data
    - Only if the ALTPCB output was from a transaction initiated from MQ or was built by the DRU exit
- The user header(s) will be passed back to the MQ application in the message on the Reply-to queue
- The format of this input message is:
  - HDR1...HDRnMQIIHLLZZTrancodeDataLLZZData...

# WebSphere MQ IMS Bridge - Input MSGs (6)

## Message Delivery Options

- Confirm On Arrival (COA) is provided when the message reaches the IMS queue
  - IMS ACKs the input message to MQ
- Confirm On Delivery (COD) is not available
  - MQ does not know when the IMS application retrieves the message from the IMS Message Queue
  - The message is rejected if this option is specified
  - There are user requirements to change this
- Expiry
  - A message can expire in MQ on the IMS Bridge Queue before being sent to IMS
    - The MQ application that MQPUT the message is notified if one of the following MQMD\_REPORT options is set
      - MQRO\_EXCEPTION (Just the Expiration report)
      - MQRO\_EXCEPTION\_WITH\_DATA (First 100 bytes of the message)
      - MQRO\_EXCEPTION\_WITH\_FULL\_DATA (All of the message)

# WebSphere MQ IMS Bridge - Input MSGs (7)

## Message Delivery Options ... cont.

- Expiry
  - MQ 7.0.1 and up supports IMS Transaction Expiration
  - MQ passes the remaining Expiry time to IMS as an IMS Transaction Expiration time
    - This is rounded up to whole seconds
    - This requires OR'ing the MQ Service Parameter with x'0000000000000001' to activate this feature
      - ZPARM CSQ6SYSP SERVICE=0000000001 + any other bits being used
      - COMMAND SYSTEM SERVICE(0000000001) + any other bits being used
  - If the transaction expires in OTMA *before* being placed on the IMS message queue it is NAK'ed by IMS (NACK\_FOR TRANS\_EXPIRED, x'0034')
    - MQ treats this as if the message had expired before being sent to IMS
    - MQMD\_REPORT options are honored
  - If the transaction expires *at the GU* to the IOPCB
    - IMS returns message DFS3688I to MQSeries
    - The DFS3688I message is returned to the Reply\_To Queue
    - The MQMD\_REPORT options are NOT honored

# WebSphere MQ IMS Bridge - Input MSGs (8)

## Message Delivery Options ... cont.

- Expiry ... cont.
    - If the transaction expires at the GU to the IOPCB  
&
    - Since WMQ enhanced the support for IMS Transaction Expiration:
      - IMS/OTMA returns the original input message to MQ instead of the DFS3688I message
      - The MQMD\_REPORT options are honored
      - The Reply message can also Expire
        - MQIHL\_FLAGS has value MQIHL\_PASS\_EXPIRATION
        - MQ will pass the REMAINING expiry time in the OTMA header
        - The reply on the reply-to queue will start the Expiry process with that remaining time
- (Any time in IMS is NOT counted)



# WebSphere MQ IMS Bridge - Input MSGs (9)

## OTMA Flood Conditions

- MQ supports (rather than tolerates) OTMA flood conditions
  - If a flood warning condition is detected the IMS Bridge will slow down sending messages to OTMA
  - The messages remain in MQ rather than flooding IMS and causing a virtual storage shortage
  - If this is an MQ Shared Queues environment another Queue Manager may be able to send the message to a different IMS

# WebSphere MQ IMS Bridge - Input MSGs (10)

## Messages sizes

- The maximum OTMA input segment size is 32,767
  - LLZZ + 32,763 bytes of data
  - IMS will create multiple records in the Large Message Queue if necessary
- The maximum OTMA total message length which can be put to the IMS Bridge Queue is the MQ maximum message length for that queue
  - This is usually 4MB
  - MQ long message support increases this to 100MB

# WebSphere MQ IMS Bridge - Input MSGs (11)

## There are special requirements for Commit Mode 0 input messages to IMS

- If the IMS transaction is defined as RECOVER then the MQ message can be persistent or non-persistent
- If the IMS transaction is NORECOV then the MQ message must be non-persistent
  - If it is persistent IMS will reject the message with sense code 00230000

# WebSphere MQ IMS Bridge - Input MSGs (12)

## There are special requirements for Commit Mode 0 input messages to IMS ... cont.

- How does IMS know whether a message is persistent or non-persistent?
  - All CM0 messages arrive on SYNChronized TPIPES
  - There are no flags in the OTMA headers for this
- The answer is that IMS and MQ use a little known but documented OTMA interface
  - If the message is persistent it is sent on the SYNChronized TPIPE with a valid sequence number
  - If the message is non-persistent it is sent on the SYNChronized TPIPE with the sequence number set to zero

# WebSphere MQ IMS Bridge - Input MSGs (13)

**If the input message cannot be put to the IMS queue because:**

- Invalid message - input message goes on DLQ and warning sent to system console
- IMS rejected message (sense 001A e.g.Transaction Stopped) - DFS message from IMS is put into MQ reply message and put on reply-to queue
  - If reply message cannot be PUT, it is placed on the DLQ
  - Input message goes to DLQ
- IMS rejected message (message error) - input message goes on DLQ and warning sent to system console
- IMS rejected message (other) - messages go back to their original queue, the IMS Bridge is stopped, and warning sent to system console

# WebSphere MQ IMS Bridge - Input MSGs (14)

**If any return messages cannot be PUT to the DLQ, messages go back to their original queue**

- If it was a queue problem the queue is stopped
- If it was an IMS Bridge problem the IMS Bridge is stopped