CICS Extreme Debugging -- MQ Attachment

Ed Addison
IBM

Thursday, August 11, 2011
Session Number 9612

Agenda

- The basic structure of the CICS-WMQ Adapter
- Differences between the CICS-shipped adapter and the WMQ-shipped adapter.
- Tuning and Monitoring implications
- WMQ and CICS dump formatters
- Dump Analysis

The CICS-MQ Adapter

- CICS/TS 3.1 does not ship the CICS-WMQ Adapter.
  - It will use the one shipped with any release of WMQ.
- CICS/TS 3.2, CICS/TS 4.1 and CICS /TS4.2 do ship the CICS-WMQ Adapter.

The WMQ-supplied Adapter

- The adapter attaches 8 subtask TCBs per CICS region. These TCBs are identified to WMQ as 'server threads.'
  - The initial program for these subtask TCBs is CSQCSERV.
- Together, these 8 'server threads' service all the WMQ API requests made by CICS tasks.
  - A CICS task doesn't own a server thread for the life of the task.
  - Many tasks within the CICS region share the 8 'server threads.'
- These 'server threads' do not maintain any affinity to the CICS tasks whose requests they service.
- They pick up a single request, a QRPL, they service it, post back the waiting CICS task, and are then free to handle the next request from any task.
- The 'server threads' live long term, usually for the life of the CICS region.

The WMQ-supplied Adapter - Continued

- 'Server threads' remain attached to their respective CICS region.
  - They are attached when the WMQ Interface is established and will remain until the Interface is terminated.
  - They do not come and go with CICS transactions.
- A WMQ thread is not the same thing as a 'server thread'. A WMQ thread aligns with a CICS task or an WMQ Unit of Recovery. It is typically short-lived.
  - A WMQ Display Thread command displays a WMQ Thread, not a 'server thread'.
  - Threads mentioned in WMQ Dump Formatters are WMQ Threads, not 'server threads.'
  - Once the CICS task is through with the WMQ call the WMQ thread returns to 'server thread' status.

The WMQ-supplied Adapter - Continued

- The WMQ CTHREAD parameter limits the number of ‘server threads.’
  - The CICS QR TCB is a special coordinator ‘server thread’
  - This coordinator server thread and the 8 server threads built on the CSQCSERV TCBs make up the 9 server threads each CICS region has.
  - Each CICS region uses up 9 of the CTHREAD limit.

The WMQ CICS - supplied Adapter

- CICS uses L8 mode TCBs for all calls to WMQ.
  - When an L8 TCB is used for a WMQ call for the first time, it is identified to WMQ as a 'server thread.'
- MAXOPENTCBS controls the number of L8 and L9 mode TCBs. If enough tasks run concurrently issuing WMQ calls, it is possible for there to be up to MAXOPENTCBS worth of 'server threads' identified to WMQ.
- An L8 TCB 'server thread' is not shared by several tasks running concurrently. For the life of a CICS task an L8 TCB 'server thread' is owned exclusively by the task for WMQ calls and for other requests that need an L8 TCB.
- The lifetime of an L8 TCB server thread is generally the life of the CICS-WMQ connection.
  - There is a decay algorithm that causes L8 TCBs to be detached if unused for an hour or so.

The WMQ CICS - supplied Adapter - Continued

- The WMQ CTHREAD parameter limits the number of ‘server threads.’
  - The CICS QR TCB is a special coordinator ‘server thread’
  - All of the L8 TCBs that have been used for a WMQ call (and therefore have been identified to WMQ) are server threads.
  - The maximum number of ‘server threads’ is MAXOPENTCBS plus 1.
- When running WMQ V6 and upgrading from CICS/TS 3.1 or lower, to CICS/TS 3.2 and above, CTHREAD may very well need to be raised.
  - Consider eliminating CTHREAD by setting to the maximum 32767.
- At WMQ V7 CTHREAD is forced to 32767 and is not adjustable.

Tuning and Monitoring Considerations

Tuning Considerations for CTHREAD

- What are the costs of having a server thread connect to a queue manager?
  - Each server thread requires about 5K from high private in the queue manager ASID, and about 5K from ECSA.
  - With WMQ V7 APAR PK69439 and WMQ V6 APAR PK68189, each server thread requires about 4K from high private in the CICS address space. Without the APARs, the 4K is from below-the-line private.

Performance Class Monitoring Considerations

- At CICS/TS 3.1 with the MQ-supplied Adapter, the CSQCSERV subtask TCBs are not managed by the CICS dispatcher.
  - CPU used by those TCBs will be charged to the CICS address space but will not be attributed to the CICS tasks whose MQ calls were serviced by those server threads.

Performance Class Monitoring implications

- At CICS/TS 3.2 and above with the CICS-supplied Adapter, the L8 Mode subtask TCBs are managed by the CICS dispatcher
  - CPU used by those server threads will be charged to the CICS address space and is attributed to the CICS task.
  - Because of this, L8CPUT and USRCPUT will be higher, especially for transactions that make lots of WMQ calls.

Inquire Task implications

- At CICS/TS 3.1 and prior, a task that is processing in WMQ would be in a TASKSWCH suspend.
- At CICS/TS 3.2 and above, a task that is processing in WMQ will be Running on an L8 TCB.
  - On EXEC CICS INQUIRE TASK, there is a CURRENTPROG attribute that returns DFHMQTRU when CICS is processing an MQ call.

WMQ and CICS Dump Formatters

VERBX DFHPDxxx ‘MQ=1’

- This MQ summary shows all the CICS tasks that are making MQ calls.
  - Note the jobname and ASID of CICS
  - Note the MQ Queue Manager name
  - Scroll down to the All Transactions Summary

VERBX DFHPDxxx ‘mq=1’

```
-- DFHPD0121I FORMATTING CONTROL BLOCKS FOR JOB CICS01


ADDRESS SPACE ASID NUMBER (HEX) =  0182
```

- The hex ASID number and Jobname will be useful when looking at the MQ formatter.

VERBX DFHPDxxx 'MQ=1'

```
==MQ: GLOBAL STATE SUMMARY

   Connection status:                    Connected
   In standby mode:                      No
   Subsys:                               CSQA
   WMQ release:                          0701
   Initiation Queue:                     CICS01.INITQ
   API Crossing exit active:             No
```

- The Queue Manager Subsystem is necessary for when invoking the MQ formatter.

VERBX DFHPDxxx 'MQ=1'

```
==MQ: ALL TRANSACTIONS SUMMARY

Tran Task  TcaAddr  TieAddr  LotAddr  ThrdAddr Uowid            Tcb
id   num                                                     in MQ
-------------------------------------------------------------------
CKTI 00044 5B705100 5CD7A1B8 5CD7A238 5CD7A290 C6EF8A4AC222C384 No
TRN1 75696 5B70E700 5D2E6650 5D2E66D0 5D2E6728 C6EFF57E75472792 Yes
TRN2 75685 5B702700 5CD7ADF8 5CD7AE78 5CD7AED0 C6EFF57E71FF8704 No
TRN2 75686 5B71C700 5D2E67D8 5D2E6858 5D2E68B0 C6EFF57E72337819 No
TRN3 75693 5B706700 5CD7A650 5CD7A6D0 5CD7A728 C6EFF57E74BF8F19 No
. . . .
```

- All of these tasks have issued WMQ calls. Only one task is currently in WMQ.
- A task showing as 'In MQ' is like a task showing in a TASKSWCH wait using the WMQ-supplied adapter.

VERBX DFHPDxxx 'DS=1'

```
S F P TT RESOURCE RESOURCE_NAME     W TIME OF       TIMEOUT       DTA       AD ATTACHER M SUSPAREA XM_TXN_TOKEN
         TYPE                         SUSPEND       DUE           (DSTSK)      TOKEN

S P N -  MQSeries GETWAIT           M 06:45:50.092      -         28B92500 XM 5D2C8300 L8 5D2E6BEC 5D2C83000075709C
S P N -  MQSeries GETWAIT           M 06:45:53.435      -         28B92680 XM 5D2D8B00 L8 5CD7AA64 5D2D8B000075720C
R                                                               28B92800 XM 5D2C3500 L8          5D2C350000075696C
. . . .
```

- Dispatcher shows this task as running
- The 'MQ=1' summary showed this task was in WMQ. Usually a task would only be in WMQ for a short duration of time.
- The times in the DS=1 summary are STCK format with no adjustments for time zone or leap seconds.
- If the task became running on its L8 TCB well before dump time, you would need to investigate that L8 TCB. What is the z/OS TCB address?
- Scroll down to the next summary in DS=1.

VERBX DFHPDxxx 'DS=1'

```
==DS: TASKS USING OPEN TCBS SUMMARY

DS_TOKEN KE_TASK  XM_TXN_TOKEN      TCB_ID DS_TCB   MVS_TCB
02800003 5B9C3100 63038D000075749C L803D  63572D00 009637D8
02820007 5BB7D900 630803000075793C L802B  638A8300 0097C658
0284047F 5BB39100 6303F3000075765C L8024  5D21E500 0098CC88
040A0071 5BB23500 5D2C350000075696C L8025  5D21EF00 0097A0A0
. . . .
```

- All of the tasks in the MQ=1 summary will also be in this summary as they will all have an L8 TCB.
- Other tasks not issuing WMQ calls may be in this summary as well.
- For task 75696, its TCB address is 97A0A0.

VERBX csqwdprd 'subsys=csqa'

- The WMQ formatter accesses control blocks in ECSA so you can do this with just a dump of CICS.
  - Use the Queue Manager Subsystem ID from CICS 'MQ=1'
  - In the output, find on the CICS hex ASID number noted earlier. Issue the find like this: f x0182

Verbx csqwdprd ‘subsys=csqa’

```
Jobname ZZC34      Conntype CICS       ASID x0182  ASCE 22D9FA48.
   EB 2189A188  ACE 2189A128  Thread 5D231238 Tran TRN1 Task 0075784C              TCB 0098C620.
   EB 21899078  ACE 21899018  Thread 5D254858 Tran TRN8 Task 0075689C              TCB 009757E0.
   EB 2189A8D8  ACE 2189A878  Thread 5D2E6548 Tran TRN8 Task 0075738C              TCB 00983AD8.
   EB 21FC5668  ACE 21FC5608  Thread 5CD7A3C0 Tran TRN7 Task 0075692C              TCB 0097A868.
   EB 21FC5530  ACE 21FC54D0  Thread 5D2313C0 Tran TRN7 Task 0075765C              TCB 0098CC88.
   EB 1F43FB48  ACE 1F43FAE8  Thread 5D2E66D0 Tran TRN1 Task 0075696C              TCB 0097A0A0.
. . .
```

- These are the current WMQ threads. These should match up with the CICS tasks in CICS ‘MQ=1’ summary.

A WMQ V6 example

```
Jobname CICSJOB1  Conntype CICS      ASID x0122.
   EB 1C962518  ACE 1C9624B8  Thread 00000000 Tran TR21 Task 0008912C
   EB 1C5733F8  ACE 1C573398  Thread 00000000 Tran TR22 Task 0058138C
   EB 1BBCFF38  ACE 1BBCFED8  Thread 00000000 Tran TR22 Task 0047300C
   EB 1BBCF998  ACE 1BBCF938  Thread 2C2EF1C8 Tran CKTI Task 0000043C
```

- There is only 1 thread, the 1 with a non-zero Thread address.
- The other 3 ‘threads’ are just sets of control blocks that used to be threads and are ready to go when needed for a new thread.
- WMQ APAR PK75212 changes the formatter to not show these free threads.

Using a Dump to View WMQ tasks

Questions to be answered

- Is CICS healthy?
- Is the task currently calling WMQ?
- How long ago was the call made?
- What program made the call?
- How can I find out the call type and locate the parameters?
- How many calls has the task made?

You've got a dump. What first?

- **Make sure CICS is healthy as a job**
  - Don't focus on task hangs if CICS itself is not healthy
  - Find out what time the dump was taken
  - Compare to CICS internal time stamps
  - If CICS is healthy those times will be close together

What time is it?

- Most timestamps are in local time
  - CICS internal trace
  - Console log
  - CICS messages
  - Kernel formatter
- Some timestamps are in GMT
  - CICS dispatcher summary
  - Units of Work (UOW)
  - Dump Incident Token time
- You need to be able to convert from GMT to local
  - Can be tricky if Leap Seconds are being used

Convert GMT time to Local

- Figure out the difference between GMT and Local
  - Issue **LTOD 0** from Option 6 of IPCS
  - This will provide the following output:

```
09/17/2042 22:53:47.370496 STCK X'00000000 00000000'
09/17/2042 22:53:47.370496 UTC X'00000000 00000000'
09/17/2042 23:53:47.370496 LOCAL X'FFFFBCF1 DCC00000
```

- In this case Local time and STCK are one hour apart exactly.
- If STCK and UTC are identical, there are no leap seconds
  - If there were Leap Seconds involved these times would be different.
  - Leap Seconds are shop dependent

What time was the dump taken?

- From option 6 of IPCS (COMMAND) issue: **ST SYS**

```
SYSTEM STATUS:
Sysplex name: SYSPLEX1
   TIME OF DAY CLOCK: C8190185 BBD100EE  07/20/2011 13:00:04.466960 local
   TIME OF DAY CLOCK: C818F41C 819100EE  07/20/2011 12:00:04.466960 GMT
   Program Producing Dump: SVCDUMP
   Program Requesting Dump: IEAVTSDT
Incident token: SYSPLEX1 MV23     07/20/2011 11:59:49.737423 GMT
```

- **Use Incident token time**
  - `11:59:49.737423` = `12:59:49.737423` **Local Time**

CSA Time-of-Day

- A byproduct of CICS being healthy is the CSA time-of-day field, CSATODP, being updated with the current time
  - When CICS is healthy, CSATODP is updated regularly
  - When CICS is unhealthy, CSATODP is not updated

- Therefore, a quick and effective way to gauge CICS's health is to compare the time the dump was taken to the CSA time-of-day field

  - CSATODP (CSA +x'50') is in the form of HHMMSSTF where
    - H is hours, M is minutes, S is seconds, T is tenths (F indicates a field in packed decimal format)
  - The CSA can be formatted using **VERBX DFHPDxxx 'CSA=2'**

Note: CSATODP is updated on every QR TCB dispatch and also when an ASKTIME is issued on ANY TCB. So, CICS can appear healthy even though the QR isn't being dispatched if another TCB (L8, L9) is issuing EXEC CICS ASKTIME

What time does CICS think it is?

- **VERBX DFHPD650 'CSA=2'**

```
0000  00000200 0004C020 0004F5A0 948E5C32  80BF3498 80800000 14FDB030 14F98260
0020  00000000 14F95100 0000010C 00000000  948E577A 15330000 14FDB030 14FC0D50
0040  00055B20 140DC800 0010032C 008B2000  1259497F 14052108 00000200 00000000
```

- **CSA +X'50' is Local Time in packed decimal field of format HHMMSST. This one is 12:59:49.7**
  - **Matches local time of dump calculated on slide 29**
  - **Updated every time a task is dispatched on the QR TCB or ASKTIME is issued on any CICS TCB**

If CSA time and Dump time are minutes apart

- CICS is probably not healthy as a job
  - CICS may be in a loop which does not update CSATODP
    - CICS will continue to loop if ICVR set to 0
    - CICS MAY continue to loop even if ICVR is set to reasonable value
  - CICS may be in a hard wait
  - CICS may be CPU starved
- Determine why CICS is not receiving resources needed to run

Are Tasks Currently calling WMQ?

- To see if a task is currently calling MQ issue VERBX DFHPDxxx 'MQ'

```
==MQ: ALL TRANSACTIONS SUMMARY
Tran Task  TcaAddr  TieAddr  LotAddr  ThrdAddr Uowid             Tcb
id   num                                                       in MQ
-------------------------------------------------------------------
CKTI 00041 140E2100 153B21B8 153B2238 153B2290 C81819C885B51E41 No
MQED 30749 140EA100 153B2340 153B23C0 153B2418 C818F40EC53D4D83 Yes
MQED 30750 140E3100 153FD7D8 153FD858 153FD8B0 C818F40EC540CB03 Yes
MQED 30751 140E9100 153FD340 153FD3C0 153FD418 C818F40EC558D9C3 Yes
MQED 30752 140E3800 153FD030 153FD0B0 153FD108 C818F40EC597E44E Yes
MQED 30753 140E4800 153FD960 153FD9E0 153FDA38 C818F40EC59A8B4E Yes
MQED 30754 140EB800 153FD650 153FD6D0 153FD728 C818F40EC5A72446 Yes
MQED 30755 140E2800 153B2DF8 153B2E78 153B2ED0 C818F40EC5B474C6 Yes
```

When was the WMQ call issued?

- Issue VERBX DFHPDxxx 'DS' to view the Dispatcher Domain

```
DS_TOKEN KE_TASK  T S RESOURCE RESOURCE TIME OF  DTA      ATTACHER M SUSPAREA XM_TXN_TOKEN
                     TYPE     NAME     SUSPEND  DSTSK)   TOKEN

040A10AD 14EEB700 N R                             13FF8800 16410B00 L8          16410B000030764C
04820001 14EFD700 N S                   19:43:17  13FF9200 14FA2300 QR 14F33A78 14FA23000000040C
04840001 14EFD100 N S MQSeries GETWAIT  11:59:50  13FF9380 14FA2500 L8 153B22BC 14FA25000000041C
048C10A7 14EFA100 N R                             13FF9980 14FA2B00 L8          14FA2B000030753C
05000C95 14ECD100 N R                             26C5C080 153F1300 L8          153F13000030749C
05020C91 14EA0100 N R                             26C5C200 1645A100 L8          1645A1000030766C
```

- Dispatcher shows this task as running on an L8 TCB
- The 'MQ=1' summary showed this task was in WMQ. Usually a task would only be in WMQ for a short duration of time.
- How long has this task been running in MQ?
- Display the DTA: ip l **26C5C080** l(X'60')

The DTA

```
LIST 26C5C080. ASID(X'0049') LENGTH(X'60') AREA
26C5C080. FFFFFFFF 05000C95 C4E2E3E2 D2C4C5C6  05000C95 C5E4C540 40404040 40404040
26C5C0A0. 26C5C080 00000000 FE000000 26C76800  FFFFFFFF 00000000 FFFFFFFF FFFFFFFF
26C5C0C0. 00000000 02FF0000 DFFFFFFF 0021FF82  C818F40E C7B8ACCE C818F40E 3B62328E
```

- For a task that is Running, DTA+X'58' is the time the task became Running, within a tenth of a second. Format this to see what time it is.
- Issue: ip ltod C818F40E3B62328E

```
07/20/2011 11:59:49.499427 STCK   X'C818F40E 3B62328E'
07/20/2011 11:59:49.499427 GTC    X'C818F40E 3B62328E'
07/20/2011 12:59:49.499427 LOCAL  X'C8190177 75A2328E'
```

- If the time is right at dump time, then the dump was taken while the task was just doing some MQ work.
- From slide 30 we found the dump was taken at 12:59:49.737423 Local Time.
- If it is some time prior, the task is hung for some reason.

What we know so far ...

- CICS Dump was taken at 12:59:49.737423 Local Time
- CSATOD (Time of Day) is 12:59:49.7 - CICS is healthy
- There currently are seven tasks in WMQ at dump time
- Task 30749 became running at 12:59:49.499427

What Program Issued the WMQ Call?

- Issue VERBX DFHPDxxx 'PG' for Program Domain and issue F 30749 for the task number

```
==PG: PTA SUMMARY FOR TRAN NUM : 30749,  PTA ADDRESS : 14DFE810
 LOG-LVL : 3          SYS-LVL : 0          TASK-LLE : 00000000  PLCB : 14ED27E8
 =PG: TASK PLCB SUMMARY
 PROG DFHMQTRU LVL   3 PLCB 14ED27E8 LD 00000000 ENT 00000000 LEN 000000 PPTE 15050E70 ENV TRUE INV MQPUTLOG EXIT
  PROGRAM: DFHMQTRU CPE: 140A6100 LIB: DFHRPL   CONCAT: 03

PROG MQPUTLOG LVL   2 PLCB 14ED12C8 LD 16448000 ENT 96448028 LEN 005B50 PPTE 153324F0 ENV EXEC INV MQGETLOC EXIT
  PROGRAM: MQPUTLOG CPE: 15331850 LIB: DFHRPL   CONCAT: 01

PROG MQGETLOC LVL   1 PLCB 14ECF060 LD 16436000 ENT 96436028 LEN 006298 PPTE 1533262O ENV EXEC INV CICS     EXIT
  PROGRAM: MQGETLOC CPE: 140AB100 LIB: DFHRPL   CONCAT: 01
```

- This display shows the first level program MQGETLOC was INVoked by CICS and LINKed to second level program MQPUTLOG.
- The second level program MQPUTLOG then made a call to DFHMQTRU. You will see this when the program is active in WMQ.
- Note that this will not show COBOL CALLs. Verification of where the call was made to WMQ is still needed.

What Program Issued the WMQ Call?

- CICS will obtain a control block called Program Environment Save Area (PESA) when stacking a user environment. This will be done when:
  - EXEC CICS LINK is issued
  - Call to a Global User Exit (GLUE) that can issue EXEC CICS commands
  - Call to a Task Related User Exit (TRUE) such as WMQ and DB2
  - Call to a User Replaceable Module (URM) such as the WMQ API Crossing Exit
- PESA can be used to get back to the program that made the WMQ call
- To find a PESA for a transaction issue VERBX DFHPDxxx ‘AP’ and then issue F PESA.xxxxx where xxxxx is the transaction number.

PESA for Transaction 30749

```
PESA.30749 14ED28A8 PROGRAM ENVIRONMENT SAVE AREA

  0000  02806EC4 C6C8D7C5 E2C10500 14ED1328  00000000 00000000 16AC3868 00000000
  0020  00000000 00000000 00000000 000016AB  E6200000 00000000 000014FA 307414ED
  0040  1F840000 0000D3F8 20080000 00000000  0006B880 80400000 00000000 000016AB
  0060  E6680000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
  0080  00000000 00000000 00000000 00001640  E0D00000 000016AC 38680000 00000000
```

- Offset X'18' for a WMQ PESA will point to Register 13 of the caller.
  - For WMQ this will be the Register Save Area when the call was made.
  - Note that offset X'96' also has the same address in case you are reviewing PESAs for other TRUEs, GLUEs or URMs.

Register Save Area when WMQ call was issued

```
16AC3868:                                 00104001   16AC36C0   |          .. ....{ |
16AC3870:   16AC3988   1644CC88           00000000   9644C502   | ...h...h....o.E. |
16AC3880:   16AC3968   166400C0           16640040   166510C0   | .......{... ...{ |
16AC3890:   166502D0   14ED1EE0           00000000   166500C0   | ...}...\.......{ |
16AC38A0:   16AC4A58   16448170           1644C3A2   16AC2990   | ..¢...a...Cs.... |
16AC38B0:   00000000   16AC3988           00000000   16AC3738   | .......h........ |
16AC38C0:   16AC3868   16AC4A58           955B1648   00000000   | ......¢.n$...... |
```

- Offset X'C' is Register 14 when the WMQ call was made and will point to the WMQ Stub.
- Offset X'14' is Register 0 when the WMQ call was made and will point to where the program actually issued the WMQ call.
- Offset X'18' is Register 1 when the WMQ call was made and will point to the parameters passed from the Application.

WMQ Stub

```
1644CC88                                47F0E008  08000005  |          .0\.....  |
1644CC90  12FF4740  E01218E0  07FE18F0  41000008            |  ... \..\...0....  |
1644CCA0  06008900  00021A10  18015811  0000D203            |  ..i...........K.  |
1644CCB0  1000E054  18104100  00041B10  58E10000            |  ..\.............  |
1644CCC0  41000002  500E0000  41000008  06000600            |  ....&...........  |
```

- Offset 4 within the WMQ stub is the number of parameters passed and the type.
  - 08 is the number of parameters passed and the type of call is 05. Most common call types are:
    - 01 = Open
    - 02 = Close
    - 03 = Get
    - 04 = Put
    - 05 = Put1

Program that made the WMQ call

- Format Loader Domain by issuing: VERBX DFHPDxxx ‘LD’
- Issue F ‘PROGRAM STORAGE MAP’ to get to the modules loaded within the CICS region
- Use the address derived from the WMQ Register Save Area on slide 40 (9644C502)

```
PGM NAME ENTRY PT  CSECT    LOAD PT. REL. PTF LVL. LAST COMPILED  COPY NO. USERS
 DFHEDFM  96402000 -noheda- 16402000                                  1        0
 MQGETLOC 96436028 DFHYI660 16436000 660                              1        18
 MQPUTLOG 96448028 DFHYI660 16448000 660                              1        8
 MQOPEN   9644F028 DFHYI660 1644F000 660                              1        5
```

- Address that made WMQ call resides in program MQPUTLOG.
  - Matches what was found in the PG domain on slide 37.
  - MQPUTLOG made the WMQ call at x’44DA’
    - 1644C502 (from PESA offset X’14’) minus program entrypoint 16448028

Parameters passed on the WMQ call

- Use register one derived from the WMQ Save Area on slide 40 (16AC3968).

```
16AC3968                          166502A0   166502F8   |          .......8 |
16AC3970   16650488   166506B0   16650290   16650684   | ...h...........d |
16AC3980   166502B8   966502D0   00000000   00000000   | ....o..}........ |
```

- There were 8 parameters passed on the PUT1 WMQ call. The list is terminated when the high order bit is on.
- The WebSphere MQ z/OS Problem Determination Guide lists all the parameters passed for all calls. Here is what's documented for MQPUT1:
  - ARG 000 Connection handle
  - ARG 001 Object descriptor
  - ARG 002 Message descriptor
  - ARG 003 Put message options
  - ARG 004 Buffer length
  - ARG 005 Message data
  - ARG 006 Completion code
  - ARG 007 Reason code

Important Arguments Passed

- ARG 001 - Object Descriptor (MQOD) will contain the Queue Name
  - 166502F8 D6C44040 00000001 | OD .... |
  - 16650300 00000001 C5C44BD3 D6C74BD8 E4C5E4C5 | ... ED.LOG.QUEUE |
- **ARG 003 – Put Message Options (MQMPO) will contain options**
  - 166506B0 D7D4D640 00000001 00000002 FFFFFFFF | PMO ..... |
    - The 00000002 indicates Syncpoint – See WMQ z/OS Problem Determination Guide for all options
- **ARG 004 – Buffer Length**
  - 16650290 00000008
- **ARG 005 – Message Data**
  - 16650684 C5C4E940 C4C1E3C1 F8F2F1F8 | EDZ DATA8218 |

What we know so far ...

- CICS Dump was taken at 12:59:49.737423 Local Time
- CSATOD (Time of Day) is 12:59:49.7 - CICS is healthy
- There currently are seven tasks in WMQ at dump time
- Task 30749 became running at 12:59:49.499427
- Program MQPUTLOG made a WMQ PUT1 call at offset X'44DA'
- WMQ PUT1 call was made to ED.LOG.QUEUE
  - SYNCPOINT Option
  - Eight byte buffer
  - Data passed was EDZ DATA

How many WMQ Calls has the task issued

- There are some helpful changes in the monitoring domain dump formatter with CICS/TS 4.1 when performance class monitoring is active.
  - For active tasks, all the non-zero monitoring fields are formatted, DFH$MOLS style.
  - There is a summary of active tasks showing a few key monitoring fields.

VERBX DFHPDxxx 'MN'

- Scroll down to the Overview and verify there are Transaction Monitoring Areas (TMA)
  - TMAs will be zero if Monitoring is not active

Number of Transaction Monitoring Areas 19

CICS Monitoring is ACTIVE
Exception Monitoring is ACTIVE
Performance Monitoring is ACTIVE
Resource Monitoring is ACTIVE
Identity Monitoring is NOT ACTIVE

VERBX DFHPDxxx 'MN'

- Find on 'TMA-DATA' to get to the new summary

```
==MN: TRANSACTION TMA-DATA SUMMARY


Tran   Tran     TMA                                                          Dispatch       Change mode
id     number   token       Start time     Dispatch time    CPU time        Suspend time     Wait time      Delay time
-------------------------------------------------------------------------------------------------------------------
CECI   0024121  153BA000    11:59:22.3794  00:00:00.2767    00:00:00.0071   Running          00:00:00.0227  00:00:00.0000
MQED   0030749  1640D000    11:59:49.4994  Running          00:00:00.0007   00:00:00.0004    00:00:00.0001  00:00:00.0001
MQED   0030750  1641D000    11:59:50.0640  Running          00:00:00.0007   00:00:00.0001    00:00:00.0000  00:00:00.0000
MQED   0030751  164C4000    11:59:50.0644  Running          00:00:00.0006   00:00:00.0002    00:00:00.0001  00:00:00.0001
MQED   0030752  153CD000    11:59:50.0652  Running          00:00:00.0007   00:00:00.0002    00:00:00.0000  00:00:00.0000
MQED   0030753  153F8000    11:59:50.0654  Running          00:00:00.0007   00:00:00.0001    00:00:00.0000  00:00:00.0000
MQED   0030754  16408000    11:59:50.0657  Running          00:00:00.0007   00:00:00.0001    00:00:00.0000  00:00:00.0000
MQED   0030755  16411000    11:59:50.0659  Running          00:00:00.0006   00:00:00.0000    00:00:00.0000  00:00:00.0000
```

VERBX DFHPDxxx 'MN'

```
==MN: TRANSACTION TMA-DATA SUMMARY


Tran   Tran     TMA                                                    Dispatch      Change mode
id     number   token      Start time    Dispatch time    CPU time     Suspend time    Wait time     Delay time
-------------------------------------------------------------------------------------------------------------
CECI   0024121  153BA000   11:59:22.3794 00:00:00.2767    00:00:00.0071  Running       00:00:00.0227 00:00:00.0000
MQED   0030749  1640D000   11:59:49.4994 Running          00:00:00.0007  00:00:00.0004 00:00:00.0001 00:00:00.0001
```

- Times are in STCK, not Local.
  - A task is always either Suspended or Dispatched.
  - Running in the Suspend Time column means the task is currently suspended. Its Suspend Time Clock is running.

VERBX DFHPDxxx 'MN'

```
==MN: TRANSACTION TMA-DATA SUMMARY


Tran   Tran     TMA                                                          Dispatch       Change mode
id     number   token       Start time     Dispatch time     CPU time      Suspend time      Wait time      Delay time
-------------------------------------------------------------------------------------------------------------------
CECI   0024121  153BA000    11:59:22.3794  00:00:00.2767  00:00:00.0071    Running         00:00:00.0227  00:00:00.0000
MQED   0030749  1640D000    11:59:49.4994  Running        00:00:00.0007    00:00:00.0004   00:00:00.0001  00:00:00.0001
```

- Running in Dispatch Time means task's Dispatch Time clock is running. It is dispatched
- In this dump, STCK time of dump is 11:59:49.737423
- Subtracting transaction start time of 30749 from the dump time shows it started 0.238 seconds ago
  - Since that task's Suspend Time is next to nothing, that means it has been Dispatched all that time. A task is always either Dispatched or Suspended.
- Scroll down below this summary
  - For each task you get a DFH$MOLS style layout of the non-zero monitoring fields so you can see what each task has done.

VERBX DFHPDxxx 'MN'

```
 MNTMA 1640D000 Transaction Monitoring Area

FIELD-NAME------------------------UNINTERPRETED--------------------------------INTERPRETED--------------

DFHTASK  C001    TRAN              D4D8C5C4                                      MQED
DFHCICS  C089    USERID            C3C9C3E2E4E2C5D9                              CICSUSER
DFHTASK  C004    TTYPE             E2C40000                                      SD..
DFHCICS  T005    START             C818F40EC527BE03                              2011/07/20 11:59:49.4994
DFHTASK  P031    TRANNUM           0030749C                                      0030749
DFHTASK  A109    TRANPRI           00000001                                             1
DFHPROG  C071    PGMNAME           D4D8C7C5E3D3D6C3                              MQGETLOC
DFHDATA  A395    WMQREQCT          00000005                                             5
DFHTASK  S007    USRDISPT          C818F40EC542244E 80000004                     Running                 4
DFHTASK  S008    USRCPUT           00000000002CE9C0 00000003                     00:00:00.000718         3
DFHTASK  S014    SUSPTIME          00000000001A664B 00000004                     00:00:00.000422         4
DFHTASK  S102    DISPWTT           000000000007474B 00000003                     00:00:00.000116         3
DFHTASK  S255    QRDISPT           0000000000045580 00000001                     00:00:00.000069         1
DFHTASK  S256    QRCPUT            00000000000467E0 00000001                     00:00:00.000070         1
```

What we know

- CICS Dump was taken at 12:59:49.737423 Local Time
- CSATOD (Time of Day) is 12:59:49.7 - CICS is healthy
- There currently are seven tasks in WMQ at dump time
- Task 30749 became running at 12:59:49.499427
- Program MQPUTLOG made a WMQ PUT1 call at offset X'44DA'
- WMQ PUT1 call was made to ED.LOG.QUEUE
  - SYNCPOINT Option
  - Eight byte buffer
  - Data passed was EDZ DATA
- Transaction 30749 has currently issued five WMQ requests

Summary

- The basic structure of the CICS-WMQ Adapter
- Differences between the CICS-shipped adapter and the WMQ-shipped adapter.
- Tuning and Monitoring implications
- WMQ and CICS dump formatters
- Dump Analysis