# How to find the date-time stamp when an MQ connection started (runmqsc DISPLAY CONN, DISPLAY CHSTATUS, plus amqldmpa)

https://www.ibm.com/support/pages/node/6207043

Date last updated: 11-May-2020

Angel Rivera – rivera@us.ibm.com
IBM MQ Support

++ Objective

Given multiple connections to a queue manager, how do you find out their chronological order from the output of “display conn” under runmqsc?
There are no time stamps shown in the output, and you want to stop the oldest of those connections.
A variation is that you want to stop the connection that has opened a queue.

The objective of this tutorial is to provide you with 2 procedures to identify the oldest connection.

a) When you issue “display conn”, the hexadecimal value given in the CONN attribute is based on the start time of the connection (plus other stuff).
You can use this information to your advantage:
If you compare the values of Connection1 and Connection2, the one with the lowest value is the Oldest!

The advantage is that it is simple to do: comparison of values.
A shortcoming is that you will not know what is the actual start time of the connection. If you want to know the time, then you will need to use the procedure in item ‘b’.

b) Given the output of “display conn” and “display chstatus”, you can select some attributes and find the values in the text output file from the MQ diagnostic tool “amqldmpa”, which will show you the start time of the connections.
A drawback is that there are many steps in the procedure and it is relatively easier to get confused.

After you have identified the desired Connection, you can stop it as follows (under runmqsc): stop conn(connectionNumberfromDisplayConn)
For more information on Connections and how to stop them, see the following tutorial:

https://www.ibm.com/support/pages/node/616249
How to identify MQ client connections and stop them

++ Note about the MQ diagnostic tool amqldmpa

The online manual mentions the utility amqldmpa only once and here is the web page:

https://www.ibm.com/support/knowledgecenter/SSFKSJ_9.1.0/com.ibm.mq.tro.doc/q114600_.htm
IBM MQ 9.1.x / IBM MQ / Troubleshooting and support /
Problem determination in DQM

IBM® MQ provides a utility to assist with problem determination named: **amqldmpa**
Your IBM service representative will provide you with the parameters you require to collect the appropriate diagnostic information...
Attention: You should not rely on the format of the output from this utility, as the format is subject to change without notice.

+ Location:

The utility is provided with the same fileset that provides the MQ Server executable commands (strmqm, endmqm, etc)

+ Linux: fileset MQSeriesServer-*

```
$ ls /opt/mqm/bin/amqldmpa
-r-sr-s--- 1 mqm mqm 15080 Mar 23 16:13 /opt/mqm/bin/amqldmpa
```

Given the full path name of a file, the following command shows which is the fileset that provides that file:

```
$ rpm -qf /opt/mqm/bin/amqldmpa
MQSeriesServer-9.1.5-0.x86_64
```

+ Windows:

C:\Program Files\IBM\MQ\bin64\amqldmpa.exe

**NOTE: For Intel hardware we need to do byte-swap for the ConnectionIDs**

We need to understand a particular behavior from **amqldmpa:**
The ConnectionIds are shown with an “end-ianess” that is appropriate for non-Intel hardware, such as AIX, Solaris SPARC, HP-UX.

For this tutorial, a Linux x86-64bit machine was used, which means that it has a different “end-ianess” and we need to do a “byte-swap”!

++ References from the online manual

The JOBNAME attribute is a useful one that we can use in conjunction with amqldmpa

https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_9.1.0/com.ibm.mq.ref.adm.doc/q086090_.htm
IBM MQ 9.1.x / IBM MQ / Reference / Administration reference / MQSC commands / DISPLAY CHSTATUS

JOBNAME
A name that identifies the MQ process that is currently providing and hosting the channel.
[UNIX, Linux, Windows, IBM i] On Multiplatforms, this name is the concatenation of the process identifier and the thread identifier of the MCA program, displayed in hexadecimal.

For more details on DISPLAY CONN see:

https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_9.1.0/com.ibm.mq.ref.adm.doc/q086140_.htm
IBM MQ 9.1.x / IBM MQ / Reference / Administration reference / MQSC commands / DISPLAY CONN

+++ Scenario

You have 2 instances of the same MQ Client application (JmsProducer) running in the same server.

+ Step 1: Baseline

At this point, there are no MQ client applications connected to the queue manager.

mqm@orizaba1.fyre.ibm.com: /home/mqm
$ runmqsc QM80

**display conn(*) where(channel NE '') APPLTAG CHANNEL CONNAME CONNOPTS**
AMQ8461: Connection identifier not found.

**display chstatus(SYSTEM.DEF.SVRCONN)**
AMQ8420: Channel Status not found.

The informational message AMQ8461 needs to be interpreted as: there are no connections. Similarly, AMQ8420 means that there are no running instances of the server-connection channel SYSTEM.DEF.SVRCONN.

+ Step 2: Starting 1st instance of the application

One instance started and it is connected successfully to a remote queue manager using a server-connection channel SYSTEM.DEF.SVRCONN.

This is the command issued at the client server:

$ **java JmsProducer -m QM80 -d Q1 -h orizaba1.fyre.ibm.com -p 1418 -l SYSTEM.DEF.SVRCONN**
Enter some text to be sent in a message <ENTER to finish>:

At the host of the queue manager, let's find out the connection details.
Notice that there are 2 connections, one connection for each session from the JMS program and this program is using 2 sessions.

$ **runmqsc QM80**

# The following shows only selected attributes.
# This query is useful to get a general idea.
# Notice that there are no time stamps!

**display conn(*) where(channel NE '') APPLTAG CHANNEL CONNAME CONNOPTS**
AMQ8276: Display Connection details.
  CONN(C5C7A95E11EDE125)

```
   EXTCONN(414D5143514D38302020202020202020)
   TYPE(CONN)
   APPLTAG(JmsProducer)                CHANNEL(SYSTEM.DEF.SVRCONN)
   CONNAME(9.46.77.213)
   CONNOPTS(MQCNO_HANDLE_SHARE_BLOCK,MQCNO_SHARED_BINDING)
AMQ8276: Display Connection details.
   CONN(C5C7A95E11EEE125)
   EXTCONN(414D5143514D38302020202020202020)
   TYPE(CONN)
   APPLTAG(JmsProducer)                CHANNEL(SYSTEM.DEF.SVRCONN)
   CONNAME(9.46.77.213)
   CONNOPTS(MQCNO_HANDLE_SHARE_BLOCK,MQCNO_SHARED_BINDING)

# Let's show all the attributes, perhaps there is one with the time stamp
# Hum! Nope! There are no time stamps.
```

**display conn(*) where(channel NE '') all**

```
AMQ8276: Display Connection details.
   CONN(C5C7A95E11EDE125)
   EXTCONN(414D5143514D38302020202020202020)
   TYPE(CONN)
   PID(17963)                          TID(13)
   APPLDESC(WebSphere MQ Channel)      APPLTAG(JmsProducer)
   APPLTYPE(USER)                      ASTATE(NONE)
   CHANNEL(SYSTEM.DEF.SVRCONN)         CLIENTID( )
   CONNAME(9.46.77.213)
   CONNOPTS(MQCNO_HANDLE_SHARE_BLOCK,MQCNO_SHARED_BINDING)
   USERID(mqm)                         UOWLOG( )
   UOWSTDA( )                          UOWSTTI( )
   UOWLOGDA( )                         UOWLOGTI( )
   URTYPE(QMGR)
   EXTURID(XA_FORMATID[] XA_GTRID[] XA_BQUAL[])
   QMURID(0.0)                         UOWSTATE(NONE)
AMQ8276: Display Connection details.
   CONN(C5C7A95E11EEE125)
   EXTCONN(414D5143514D38302020202020202020)
   TYPE(CONN)
   PID(17963)                          TID(13)
   APPLDESC(WebSphere MQ Channel)      APPLTAG(JmsProducer)
   APPLTYPE(USER)                      ASTATE(NONE)
   CHANNEL(SYSTEM.DEF.SVRCONN)         CLIENTID( )
   CONNAME(9.46.77.213)
   CONNOPTS(MQCNO_HANDLE_SHARE_BLOCK,MQCNO_SHARED_BINDING)
   USERID(mqm)                         UOWLOG( )
   UOWSTDA( )                          UOWSTTI( )
   UOWLOGDA( )                         UOWLOGTI( )
```

```
   URTYPE(QMGR)
   EXTURID(XA_FORMATID[] XA_GTRID[] XA_BQUAL[])
   QMURID(0.0)                    UOWSTATE(NONE)

# OK, now let's show the channel status (summary)

display chstatus(SYSTEM.DEF.SVRCONN)
AMQ8417: Display Channel Status details.
   CHANNEL(SYSTEM.DEF.SVRCONN)          CHLTYPE(SVRCONN)
   CONNAME(9.46.77.213)                 CURRENT
   STATUS(RUNNING)                    SUBSTATE(RECEIVE)

# Hum, there are no time stamps and no ConnectionIDs!

# Let's invoke a more complete output by including the specification: all

# Notice we can see the Date and the Time when the instance was started:
   CHSTADA(2020-05-07)                CHSTATI(10.23.20)

# Hum, but we do not see what is the ConnectionId!

display chstatus(SYSTEM.DEF.SVRCONN) all
AMQ8417: Display Channel Status details.
   CHANNEL(SYSTEM.DEF.SVRCONN)          CHLTYPE(SVRCONN)
   BUFSRCVD(21)                       BUFSSENT(20)
   BYTSRCVD(3764)                      BYTSSENT(4360)
   CHSTADA(2020-05-07)                  CHSTATI(10.23.20)
   COMPHDR(NONE,NONE)                    COMPMSG(NONE,NONE)
   COMPRATE(0,0)                      COMPTIME(0,0)
   CONNAME(9.46.77.213)                 CURRENT
   EXITTIME(0,0)                      HBINT(300)
   JOBNAME(0000462B0000000D)              LOCLADDR(::ffff:9.46.77.213(1418))
   LSTMSGDA(2020-05-07)                LSTMSGTI(10.23.21)
   MCASTAT(RUNNING)                    MCAUSER(mqm)
   MONCHL(OFF)                        MSGS(16)
   RAPPLTAG(JmsProducer)                 SECPROT(NONE)
   SSLCERTI( )                        SSLKEYDA( )
   SSLKEYTI( )                        SSLPEER( )
   SSLRKEYS(0)                         STATUS(RUNNING)
   STOPREQ(NO)                          SUBSTATE(RECEIVE)
   CURSHCNV(2)                          MAXSHCNV(10)
   RVERSION(08000009)                    RPRODUCT(MQJM)
```

+ Step 3: Starting 2nd instance of the application

Another instance started today and also connected successfully (same remote queue manager, same channel).

This is the command issued at the client server:

$ **java JmsProducer -m QM80 -d Q1 -h orizaba1.fyre.ibm.com -p 1418 -l SYSTEM.DEF.SVRCONN**
Enter some text to be sent in a message <ENTER to finish>:

At the host of the queue manager, let's find out the connection details.

$ **runmqsc QM80**

# Keep in mind that JmsProducer needs 2 connections!
# Because we have now 2 instances, and 1 instance uses 2 connections, we have a total of
# 4 connections

**display conn(*) where(channel NE '') APPLTAG CHANNEL CONNAME CONNOPTS**

# These are the original 2 connections (for the 1st instance in Step 2)

```
AMQ8276: Display Connection details.
   CONN(C5C7A95E11EDE125)
   EXTCONN(414D5143514D38302020202020202020)
   TYPE(CONN)
   APPLTAG(JmsProducer)                CHANNEL(SYSTEM.DEF.SVRCONN)
   CONNAME(9.46.77.213)
   CONNOPTS(MQCNO_HANDLE_SHARE_BLOCK,MQCNO_SHARED_BINDING)
AMQ8276: Display Connection details.
   CONN(C5C7A95E11EEE125)
   EXTCONN(414D5143514D38302020202020202020)
   TYPE(CONN)
   APPLTAG(JmsProducer)                CHANNEL(SYSTEM.DEF.SVRCONN)
   CONNAME(9.46.77.213)
   CONNOPTS(MQCNO_HANDLE_SHARE_BLOCK,MQCNO_SHARED_BINDING)
```

# Thus, these are the 2 new connections (for the 2nd instance)

```
AMQ8276: Display Connection details.
   CONN(C5C7A95E11EFE125)
   EXTCONN(414D5143514D38302020202020202020)
   TYPE(CONN)
   APPLTAG(JmsProducer)                CHANNEL(SYSTEM.DEF.SVRCONN)
   CONNAME(9.46.77.213)
   CONNOPTS(MQCNO_HANDLE_SHARE_BLOCK,MQCNO_SHARED_BINDING)
```

```
AMQ8276: Display Connection details.
   CONN(C5C7A95E11F0E125)
   EXTCONN(414D5143514D38302020202020202020)
   TYPE(CONN)
   APPLTAG(JmsProducer)                CHANNEL(SYSTEM.DEF.SVRCONN)
   CONNAME(9.46.77.213)
   CONNOPTS(MQCNO_HANDLE_SHARE_BLOCK,MQCNO_SHARED_BINDING)

# Let's show the channel status

# Notice that now we have 2 entries in the output!

# The first entry has a startup of:
   CHSTADA(2020-05-07)                 CHSTATI(10.23.20)

# The second entry has a startup of:
   CHSTADA(2020-05-07)                 CHSTATI(10.35.42)

display chstatus(SYSTEM.DEF.SVRCONN) all

AMQ8417: Display Channel Status details.
   CHANNEL(SYSTEM.DEF.SVRCONN)         CHLTYPE(SVRCONN)
   BUFSRCVD(23)                        BUFSSENT(22)
   BYTSRCVD(3820)                      BYTSSENT(4416)
   CHSTADA(2020-05-07)                 CHSTATI(10.23.20)
   COMPHDR(NONE,NONE)                  COMPMSG(NONE,NONE)
   COMPRATE(0,0)                       COMPTIME(0,0)
   CONNAME(9.46.77.213)                CURRENT
   EXITTIME(0,0)                       HBINT(300)
   JOBNAME(0000462B0000000D)           LOCLADDR(::ffff:9.46.77.213(1418))
   LSTMSGDA(2020-05-07)                LSTMSGTI(10.23.21)
   MCASTAT(RUNNING)                    MCAUSER(mqm)
   MONCHL(OFF)                         MSGS(16)
   RAPPLTAG(JmsProducer)               SECPROT(NONE)
   SSLCERTI( )                         SSLKEYDA( )
   SSLKEYTI( )                         SSLPEER( )
   SSLRKEYS(0)                         STATUS(RUNNING)
   STOPREQ(NO)                         SUBSTATE(RECEIVE)
   CURSHCNV(2)                         MAXSHCNV(10)
   RVERSION(08000009)                  RPRODUCT(MQJM)

AMQ8417: Display Channel Status details.
   CHANNEL(SYSTEM.DEF.SVRCONN)         CHLTYPE(SVRCONN)
   BUFSRCVD(20)                        BUFSSENT(19)
   BYTSRCVD(3736)                      BYTSSENT(4332)
   CHSTADA(2020-05-07)                 CHSTATI(10.35.42)
```

```
   COMPHDR(NONE,NONE)                COMPMSG(NONE,NONE)
   COMPRATE(0,0)                   COMPTIME(0,0)
   CONNAME(9.46.77.213)              CURRENT
   EXITTIME(0,0)                   HBINT(300)
   JOBNAME(0000462B0000000E)           LOCLADDR(::ffff:9.46.77.213(1418))
   LSTMSGDA(2020-05-07)              LSTMSGTI(10.35.42)
   MCASTAT(RUNNING)                MCAUSER(mqm)
   MONCHL(OFF)                     MSGS(16)
   RAPPLTAG(JmsProducer)             SECPROT(NONE)
   SSLCERTI( )                     SSLKEYDA( )
   SSLKEYTI( )                     SSLPEER( )
   SSLRKEYS(0)                      STATUS(RUNNING)
   STOPREQ(NO)                       SUBSTATE(RECEIVE)
   CURSHCNV(2)                       MAXSHCNV(10)
   RVERSION(08000009)                RPRODUCT(MQJM)
```

Step 4: OK, we have reached an apparent impasse in our procedure:

In order to stop a connection, we need to know the connection id, and we get the connection id by issuing:

**display conn(*) where(channel NE '') APPLTAG CHANNEL CONNAME CONNOPTS**

The connection id is shown in the attribute CONN:

```
AMQ8276: Display Connection details.
   CONN(C5C7A95E11EDE125)
   EXTCONN(414D5143514D38302020202020202020)
```

In this example, the connection id would be:
   C5C7A95E11EDE125

This is fine when there is only 1 connection.

But when there are more connections, which is the one that started first?

Let’s show the 4 CONN entries from our output:

```
CONN(C5C7A95E11EDE125)  => this is the oldest (lowest value)
CONN(C5C7A95E11EEE125)
CONN(C5C7A95E11EFE125)
CONN(C5C7A95E11F0E125)  => this is the youngest (highest value)
```

The ConnectionID is obtained by combining several elements, and one of them is the date and time when the connection started. Thus, a lower hexadecimal value indicates that it

was an earlier connection and a higher hexadecimal value indicates a more recent connection.

Step 4.5) Additional information

Another way to look at the data is to find out the JOBNAME for each of the instances of the server-connection channel:

```
AMQ8417: Display Channel Status details.
   CHANNEL(SYSTEM.DEF.SVRCONN)          CHLTYPE(SVRCONN)
   CHSTADA(2020-05-07)                  CHSTATI(10.23.20)
   JOBNAME(0000462B0000000D)

AMQ8417: Display Channel Status details.
   CHANNEL(SYSTEM.DEF.SVRCONN)          CHLTYPE(SVRCONN)
   CHSTADA(2020-05-07)                  CHSTATI(10.35.42)
   JOBNAME(0000462B0000000E)
```

From the above, we can see that
JOBNAME(0000462B0000000D) is the OLDEST the time stamp is CHSTATI(10.23.20)

Step 5) Finding out the actual start date and time for a connection

So far, we are dealing with a small number of connections and we are cheating a little bit because of the different snapshots from runmqsc we know which connection is first and which is last!

But, what do you do if you have 50 connections and they been running for several days?

Well, that is when we can use the MQ diagnostic utility “amqldmpa” as follows:
General format:

**amqldmpa -c K -d 8 -m <qmgr> -f <output file name>**

For this queue manager, let’s issue the following. Notice that when it is successful, it just returns the original prompt.

```
mqm@orizaba1.fyre.ibm.com: /home/mqm
$ amqldmpa -c K -d 8 -m QM80 -f /tmp/amqldmpa.QM80.txt
amqldmpa -c K -d 8 -m QM80 -f /tmp/amqldmpa.QM80.txt
mqm@orizaba1.fyre.ibm.com: /home/mqm
```

Then, we will have to review the output file:

```
$ vi /tmp/amqldmpa.QM80.txt
```

Earlier we identified that the following JOBNAME is the OLDEST:
JOBNAME(0000462B0000000D) is the OLDEST the time stamp is CHSTATI(10.23.20)

The value for JOBNAME, on Multiplatforms, this name is the concatenation of the process identifier and the thread identifier of the MCA program, displayed in hexadecimal.

Let's proceed to findout the DECIMAL numbers for the Process Id and the Thread Id.

One site that provides a hexadecimal to decimal conversion is:

https://www.rapidtables.com/convert/number/hex-to-decimal.html

```
Provided the first 8 bytes in hex from JOBNAME:
  0000462B0000000D  => 0000462B   +   0000000D
                       ProcessID      ThreadID

That is, the hex for the ProcessID is:
  0000462B
The conversion to Decimal is:
  17963
.
That is, the hex for the ThreadID is:
  0000000D
The conversion to Decimal is:
  13

It means that the JOBNAME refers to:
  Connection(ProcessID.ThreadID)
In this case:
  Connection(17963.13)

Let's search the output from amqldmpa for the above.
NOTICE! There could be several entries in the file, thus, after finding the 1st instance,
you need to keep looking!
In this example, we have 2 entries

NOTICE: That the MQ queue manager process that is handling this connection is:
  ApplPid:                17963
  OrigApplName:           amqrmppa

Let's show the runtime details of the process 17963

mqm@orizaba1.fyre.ibm.com: /home/mqm
$ ps -ef | grep 17963
mqm      17963 17722  0 Apr29 ?        00:00:33 /opt/mqm80/bin/amqrmppa -m QM80
```

```
Let’s resume our review of the output file from amqldmpa:

# This is the first entry

Connection(17963.13)
{
  ConnectionId:           5EA9C7C5 25E1ED11
  ConnSeqNo:              00000000 000001BF
  ApplPid:                17963
  ApplTid:                13
  AgentPid:               17690
  AgentTid:               26
  connectTime:            2020-05-07 10:23:20.951
  ConnectOptions:         HandleShareBlock|SharedBinding
  ConnectionState:        2
                          kqiCONNSTATE_ADOPTED_USER
  PrivilegeOptions:       4084001
  MaxMsgLength:           4194304
  AuthToken:              1589276992
  bAlternateIdSet:        TRUE
  ApplName:               JmsProducer
  ApplType:               MQAT_JAVA
  AccountingToken:        00000000 00000000 00000000 00000000
                          00000000 00000000 00000000 00000000
  UserContext.UserId      mqm
  OrigApplName:           amqrmppa
  OrigApplType:           MQAT_QMGR
  UOWSequence:            1
  ChannelName:            SYSTEM.DEF.SVRCONN
  ConnectionName:         9.46.77.213
  ProductIdentifier:      MQJM08000009
  AppConnector             01::00::00-01083072
  AppConnId                110
  PrivilegeApplType:      36
  ApplDesc:               'WebSphere MQ Channel
'
}

This is the 2nd entry.
Notice that it has a handle for queue Q1

Connection(17963.13)
{
  ConnectionId:           5EA9C7C5 25E1EE11
  ConnSeqNo:              00000000 000001C0
  ApplPid:                17963
  ApplTid:                13
  AgentPid:               17690
  AgentTid:               27
  connectTime:            2020-05-07 10:23:21.042
  ConnectOptions:         HandleShareBlock|SharedBinding
  ConnectionState:        2
                          kqiCONNSTATE_ADOPTED_USER
  PrivilegeOptions:       4084001
  MaxMsgLength:           4194304
```

```
  AuthToken:              1589294955
  bAlternateIdSet:        TRUE
  ApplName:               JmsProducer
  ApplType:               MQAT_JAVA
  AccountingToken:        00000000 00000000 00000000 00000000
                          00000000 00000000 00000000 00000000
  UserContext.UserId      mqm
  OrigApplName:           amqrmppa
    OrigApplType:           MQAT_QMGR
    UOWSequence:            1
    ChannelName:            SYSTEM.DEF.SVRCONN
    ConnectionName:         9.46.77.213
    ProductIdentifier:      MQJM08000009
    AppConnector             01::00::00-01101056
    AppConnId                112
    PrivilegeApplType:      36
    ApplDesc:               'WebSphere MQ Channel
'
    Hobj
    {
      StrucId                 KHOB
      Self:                   02::05::05-18006144
      Status:                 0x7
      ObjectType:             1
      ObjectName:             Q1
      ObjectQMgrName:         QM80
      OpenOpts:               MQOO_OUTPUT|MQOO_FAIL_IF_QUIESCING
      {
        StrucId                 KQPE
        mbQHandle:              02::11::11-01995520
        pQHandle                0x7feb711b6300
      }QPath[0]
      mbQHandle:              02::11::11-01995520
      HState:                 0
      qcQueueStatus           02::05::05-18173184
      fOtherQmgr:             1
      xcsConnId               81
    }
  }
```

The values for the ConnectionID for both entries are shown below.

```
  ConnectionId:           5EA9C7C5 25E1ED11
  ConnectionId:           5EA9C7C5 25E1EE11
```

At first sight, we could just construct the full ConnectionID by concatenating the bytes:

```
  ConnectionId:           5EA9C7C5 25E1ED11 => 5EA9C7C525E1ED11
```

But if we try to use runmqsc and issue ‘display CONN’, it will not be found!

```
$ runmqsc QM80
display CONN(5EA9C7C525E1ED11)
AMQ8461: Connection identifier not found.
```

Because the example here was taken from an Intel box, the bytes that are displayed are

"reversed".

Starting with the following:

  ConnectionId:           5EA9C7C5 25E1ED11

We need to "un-reverse" them by doing the following swapping for each set of 8 bytes:

  5EA9C7C5

Add a blank space to separate each pair (for easy identification)

  5E A9 C7 C5

Now swap the bytes:

```
  5E A9 C7 C5 =>  C5 C7 A9 5E
           ++ --- ++
        ++ --------- ++
     ++ --------------- ++
  ++ --------------------- ++
```

The result of the swap is:
  C5 C7 A9 5E

Let' s remove the extra spaces from the result, which gives:
  C5C7A95E

Let's repeat the process for the other set of 8 bytes:

25E1ED11

Add spaces:
25 E1 ED 11

Swap:
25 E1 ED 11 => 11 ED E1 25

Result is:
11 ED E1 25

Remove spaces:
11EDE125

OK, now we have:
 C5C7A95E  +  11EDE125

Let's concatenate them:
        C5C7A95E11EDE125

This is the value for CONN ...
   display CONN(C5C7A95E11EDE125)

And now it worked!

```
display CONN(C5C7A95E11EDE125)
AMQ8276: Display Connection details.
   CONN(C5C7A95E11EDE125)
   EXTCONN(414D5143514D38302020202020202020)
   TYPE(CONN)
   PID(17963)                              TID(13)
   APPLDESC(WebSphere MQ Channel)          APPLTAG(JmsProducer)
   APPLTYPE(USER)                          ASTATE(NONE)
   CHANNEL(SYSTEM.DEF.SVRCONN)             CLIENTID( )
   CONNAME(9.46.77.213)
   CONNOPTS(MQCNO_HANDLE_SHARE_BLOCK,MQCNO_SHARED_BINDING)
   USERID(mqm)                             UOWLOG( )
   UOWSTDA( )                              UOWSTTI( )
   UOWLOGDA( )                             UOWLOGTI( )
   URTYPE(QMGR)
   EXTURID(XA_FORMATID[] XA_GTRID[] XA_BQUAL[])
   QMURID(0.0)                             UOWSTATE(NONE)
```

+++ end