Accessing Enterprise Transactions from Lotus Domino Using MQSeries

An IBM e-business Experience Report

Contents

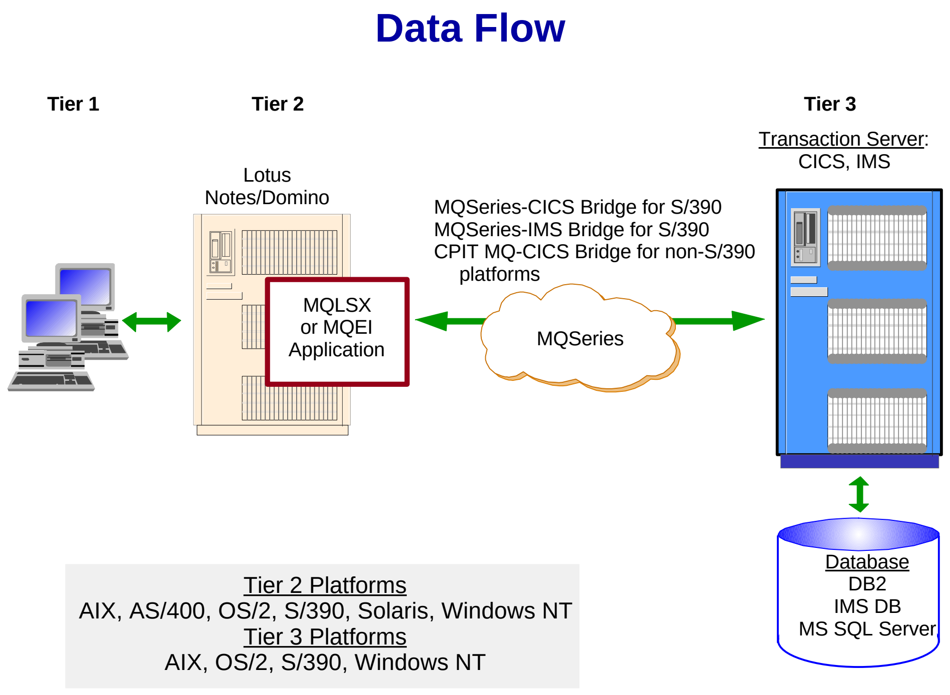
A Quick Look

Business Problem

Our pseudo customer was a large enterprise with multiple sites. The customer's enterprise had grown over the years through expansion and acquisition. This had left the customer with many disparate systems and platforms that must be integrated. The customer had settled on Lotus Notes for enterprise groupware and messaging. Business logic had been developed based on Lotus Notes and that logic now required access to enterprise data embodied in several Tier 3 transaction systems. These enterprise data platforms were predominantly IBM S/390 with some AIX, Windows NT, and OS/2. The enterprise transaction systems and databases included CICS, IMS, DB2, and Microsoft SQL Server.

Solution

We generated a single solution that supported multiple platforms on Tier 2 and Tier 3. We created two implementations of this solution. One implementation used MQSeries link LotusScript Extension (MQLSX) on Tier 2; the other used MQSeries Enterprise Integrator (MQEI). Both implementations used MQSeries to communicate between Tier 2 and Tier 3. The general data flow is demonstrated in the following figure.


Data Flow
Tier 1
Tier 2
Tier 3
Transaction Server: CICS, IMS
Lotus Notes/Domino
MQLSX or MQEI Application
MQSeries-CICS Bridge for S/390
MQSeries-IMS Bridge for S/390
CPIT MQ-CICS Bridge for non-S/390 platforms
MQSeries
Database
DB2
IMS DB
MS SQL Server
Tier 2 Platforms
AIX, AS/400, OS/2, S/390, Solaris, Windows NT
Tier 3 Platforms
AIX, OS/2, S/390, Windows NT

Characteristics

- Tier 3 transactions systems included the following:
  - CICS and DB2 on S/390
  - IMS and IMS DB on S/390
  - CICS and DB2 on AIX
  - CICS and DB2 on OS/2

    - CICS and DB2 on Windows NT
    - CICS and Microsoft SQL Server on Windows NT
- Tier 2 systems included the following:
    - S/390
    - AIX
    - AS/400
    - OS/2
    - Solaris
    - Windows NT
- We used the IBM MQSeries-IMS Bridge and the IBM MQSeries-CICS Bridge programs to interface between MQSeries and the existing transaction systems on S/390.
- To maintain a common solution across all platforms we wrote a simplified, demonstration bridge to interface between MQSeries and CICS on AIX, OS/2, and Windows NT. We refer to this as the CPIT MQSeries-CICS Bridge.
- Our solution involved systems at four sites: Austin, TX; Dallas, TX; Poughkeepsie, NY; and Rochester, MN.

Discoveries

- **Common solution for multiple platforms**
  We were able to implement a single solution across all the platforms specified above.
- **Getting IBM CICS on Windows NT to work with MS SQL Server was difficult**
  The current IBM documentation is misleading. We were able to get it to work after much effort. The documentation and sample programs should be clarified in the next release.
- **There is no CICS-MQSeries Bridge for non-S/390 platforms**
  IBM supplies the MQSeries-CICS Bridge for S/390 but there is no similar product to provide the interface between MQSeries and CICS for other platforms. We implemented a simple program to provide this function (without the robustness of the S/390 product).
- **ReplyToQueue not allowed on Tier 3 when using MQEI**
  When setting up the Tier 2 to Tier 3 MQEI configuration, we discovered that the use of a ReplyToQueue on Tier 3 is not allowed. This is not clearly documented.
- **Converting a LotusScript agent from Notes forms to Web based is difficult**
  We developed a LotusScript agent which uses Notes forms to execute CICS transactions on S/390 from Domino on Tier 2. The agent uses MQLSX on Tier 2 to interface to the MQSeries-CICS Bridge for S/390 to accomplish this. We found it quite difficult to convert this agent to a Web based agent. It required new conversion tables and specifying use of a different character set for the Web based agent which was not clear from the documentation.
- **Converting an existing CICS 3270-driven transaction to use the MQSeries-CICS Bridge is difficult**
  Taking a CICS 3270-driven transaction and breaking it into separate client and server programs, then substituting the MQSeries-CICS Bridge in place of the client program, required digging through several different manuals and not everything we had to do was documented. Getting the MQSeries-CICS Bridge to start our application program and have it assigned the appropriate transaction ID, which we could identify with a DB2 plan to allow us to connect to DB2, was not documented. It was only after we got it working that MQSeries development confirmed we had made the right assumptions and that we had set it up as they

intended. Having a place where this entire procedure appears should save customers a lot of time and effort.

- **Limited debugging facilities**
  We found that debugging MQLSX problems on S/390 was very difficult due to the lack of messages and tracing/debugging tools. This RAS issue is being evaluated to see if it will be included in a future release of Domino for S/390.

Implementation Details

Overview

Business Problem and Solution

Problem

Our pseudo customer was a large enterprise with multiple sites. The customer's enterprise had grown over the years through expansion and acquisition. This had left the customer with many disparate systems and platforms that must be integrated. The customer had settled on Lotus Notes for enterprise groupware and messaging. Business logic had been developed based on Lotus Notes and that logic now required access to enterprise data embodied in several Tier 3 transaction systems. These enterprise data platforms were predominantly IBM S/390 with some AIX, Windows NT, and OS/2. The enterprise transaction systems and databases included CICS, IMS, DB2, and Microsoft SQL Server.

This report describes our experiences in developing a solution to this problem using Lotus Notes LotusScript extensions (MQLSX and MQEI) and MQSeries. The experiences presented here are representative of what would be encountered by a real customer that was developing this same type of solution. However, our experiences do not cover all possibilities for such a solution. It is our intent to share our experiences to help you make informed decisions about your own environment and make your implementation go more smoothly.

Solution

This pseudo customer formed the basis for our decisions regarding what to include in our testing. We used MQLSX and MQEI (LotusScript extensions) and MQSeries along with Domino to interface with legacy transaction systems on the host. The network connectivity employed was TCP/IP. We used the following three-tier model

- Tier One - Web browsers, Notes clients
- Tier Two - Domino servers, MQSeries
- Tier Three - MQSeries, CICS, DB2, IMS, Microsoft SQL Server

Within this environment we executed the following scenarios:

- Domino using MQLSX to enterprise transaction systems via MQSeries
- Domino using MQEI to enterprise transaction systems via MQSeries
- Domino using MQLSX to enterprise IMS transaction systems via MQSeries
- Domino using MQEI to enterprise IMS transaction systems via MQSeries

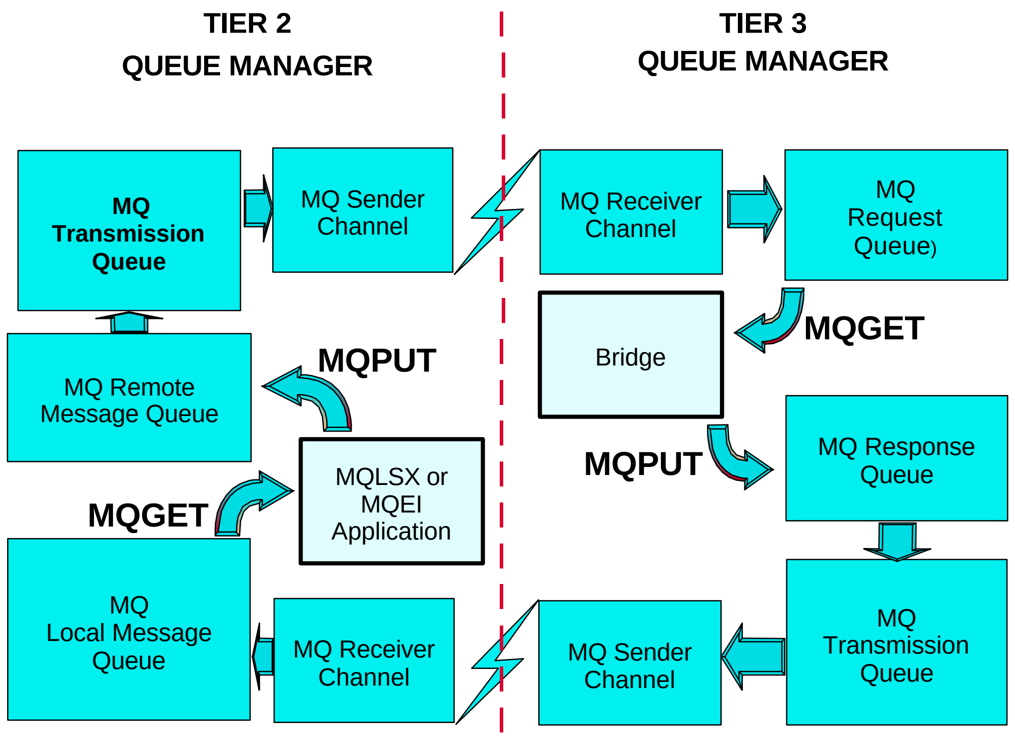
Environment Highlights

Objectives

Our objective was to demonstrate a common solution to accessing legacy data, via existing transactions, from business logic residing on a Lotus Domino Server. We intended to demonstrate a common solution across several platforms using IBM product content whenever possible.

Basic Scenario

Our scenario involved developing applications based on the following IBM products: IBM DB2 Universal Database, IBM MQSeries, IBM CICS Transaction Server, and Lotus Domino Server. We built applications using LotusScript agents written in MQSeries link LotusScript Extension (MQLSX) and MQSeries Enterprise Integrator (MQEI). We used the IBM MQSeries-IMS Bridge and the IBM MQSeries-CICS Bridge programs to interface between MQSeries and the existing transaction systems on S/390. To maintain a common solution across all platforms we wrote our own simplified, demonstration bridge to interface between MQSeries and CICS on AIX, OS/2, and Windows NT. We refer to this as the CPIT MQSeries-CICS Bridge. (See Appendix C for a description and source code.) Our solution is a three tier implementation. Our MQLSX and MQEI applications are used to simulate business logic on Tier 2 and are not complete implementations of business logic solutions. The basic flow is illustrated in the following figure and described below.



Process Flow Through the Solution Components (Using CICS and DB2 on Tier 3)

The process flow through these components is summarized below:

1. Either a Notes Form or an Internet browser is used to initiate submission of a message from an agent on the Tier 2 MQ Server to the Tier 3 system.
2. On the Tier 3 system, the bridge application picks up the message and, if it is properly formatted, sends it to the transaction system for processing.
3. Upon completion of the transaction, the bridge formats the results as an MQ message and places it the appropriate Response Queue(Note 1).
4. From the Response Queue, the message is sent to the Transmission Queue and finally back to the appropriate Tier 2 system.
5. The agent displays the results in proper format for the Notes Form or Internet browser.

Notes:

1. When running the MQEI agent, the Response Queue on Tier 3 must not be configured. The message is put directly on the Transmission Queue.

Testing Experiences

In order to make our report more useful to you, we have separated the installation and run-time experiences by platform. Common experiences, such as application development, have their own sections. Our run-time experiences include the following platforms:

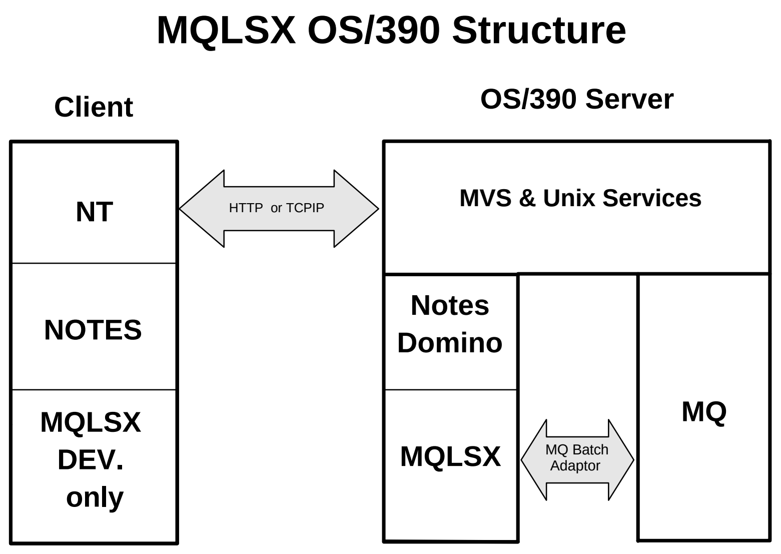
- Tier 2 systems: S/390, AIX, AS/400, OS/2, Solaris, Windows NT
- Tier 3 transactions systems: CICS and DB2 on S/390, IMS on S/390, CICS and DB2 on AIX, CICS and DB2 on OS/2, CICS and DB2 on Windows NT, CICS and Microsoft SQL Server on Windows NT

Experiences with the S/390 platform

MQSeries link LotusScript Extension (MQLSX)

Overview
MQLSX gives your Lotus Notes applications, written in LotusScript, the ability to communicate with applications running in the OS/390 environment using the Lotus Notes Domino Server on the OS/390 platform. This integration between Lotus Notes and MQSeries software extends the scope and power of Notes to include data and transactions that are part of the OS/390 environment.

The following diagram depicts an overview of the MQLSX for OS/390 model.



No Need for MQ on the client, No need for client attached facility for MQSeries on S/390

Poughkeepsie S/390 environment:
- MVS Rel 2.7
- MQ V1R2
- Domino 4.6.3
- MQLSX rel 1.3.1

Packaging
The S/390 MQLSX product has two parts to install:

- S/390 server
- Windows NT client

To download the required code from the Internet, we used the following sites:
- S/390 MQLSX server located at http://www.software.ibm.com/ts/mqseries/txppacs/ma1h.html
- MQLSX NT client located at http://www.software.ibm.com/ts/mqseries/txppacs/ma7d.html

Installation

S/390 MQLSX server installation

- Download MQLSX for S/390 server and the Windows NT client support code to a temporary directory
- Create HFS data set, we called it: **omvsspa.lotus.mqlsx.hfs**
- Create HFS mounting point in the root directory, to mount the HFS data set: **/mqm**
- Mount your dataset to the HFS mounting point, using either ISH options for mounting the HFS, or use OE mount command. We used the ISH options (**file_systems -- mount**)
- From the root directory, use the tar command to untar the downloaded file: **tar -xpozf mqlsxos3.tar.z**
- The tar command will create:
  /usr/lpp/lotus/notes/latest/os390/libmqlsx
  /usr/lpp/lotus/notes/latest/os390/libmqlsx.x
  /mqm/mqlsx/readme
  /mqm/mqlsx/conv/*.tbl
  /notesdata/mqlsxdoc.nsf
  /notesdata/mqlsxverify.nsf
- Read the **/mqm/mqlsx/readme** file and open **/notesdata/mqlsxdoc.nsf** database, follow the instructions given**.**
- Modify your user profile to include the following environment variables:
  export GMQ_TRACE=on
  export GMQ_TRACE_PATH=/mqm/mqlsx/trace
  export GMQ_TRACE_LEVEL=9
  export GMQ_XLAT_PATH=/mqm/mqlsx/table
- Modify the step libs in user profile to include the following environment MQ load libs data sets:
  STEPLIB=MQS.LOCAL.SCSQAUTH:MQS.V1R2M0.SCSQAUTH:MQS.V1R2M0.SCSQLOAD
- Add the following to **/notesdata/notes.ini**:
  AMGR_DOCUPDATEAGENTMININTERVAL=1
  AMGR_DOCUPDATEEVENTDELAY=1
  LOG_AGENTMANAGER=1

MQLSX Windows NT Client for S/390 installation

- Unzip the **MA7D.zip** file in the **c:\**
  \utility\pkunzip -d a:\MA7D.zip c:\mqm

- Modify config file as follows
  - Control panel >> system >> environment
    Add the following system variable: **C:\mqm\MQLSX\conv**
    Add the following user variable: **MQLSX --> c:\mqm\MQLSX\bin**
- Reboot your workstation
- Add the "**mqlsxverify.nsf**" database to desktop

<u>Verification and criteria for success:</u>
In order to verify the successful installation of the MQLSX software, bring up the S390 domino server and, on the NT domino client, perform the following actions:

- Open the MQLSX Install Verify database
- Go to the agent >> set the S390 server name and save the document
- Go to the verify view >> use the "create" >> Verify pull down
  Type your MQ mgr
- Save the document will trigger the agent.
- Then check the Agent Log and/or the server console for following messages. **If everything is correct, you should see all 26 messages.**
  1. Verification program has started and we have successfully loaded MQLSX DLL
  2. Successful dim on the MQ objects
  3. Successful New MQSession
  4. Successful New MQQueueManager
  5. Successful New MQPutMessageOptions
  6. Successful New MQGetMessageOptions
  7. Successful New MQMessage
  8. Successful Connect
  9. Successfully AccessQueue SYSTEM.DEFAULT.LOCAL.QUEUE
  10. Successful WriteString Hello World
  11. Successful setting MessageType to MQMT_DATAGRAM
  12. Successful setting Format to MQMT_STRING
  13. Queue depth before put: 0
  14. Successful putting the message
  15. After PUT and COMMIT queue depth is: 1
  16. Successful setting MQGetMessageOptions
  17. Queue depth before get: 1
  18. Success getting the message
  19. Successful ReadString
  20. Message put matches message got
  21. Message Put:..Hello World ..
  22. Message Got:..Hello World ..
  23. Successful Committing the message
  24. After Get and COMMIT queue depth is: 0
  25. Successful Disconnecting
  26. Verification Routine has completed. Your install is successful if you have no errors.

<u>Hints & Tips</u>

- During the verification of MQLSX installation, the CPIT Poughkeepsie site encountered an environment problem. We received the following messages when we tried to run the MQLSX verify agent.

  ```
  04/02/99 07:06:25     AMgr: Agent 'MQLSX OS/390 install verification' in
  'mqlsxverify.nsf' did not process all documents successfully.  Check the
  Agent Log for more information: Unknown LotusScript Error.
  ```

  We wrote Problem number Re: Sev1 PMR 74536,994238 with MQSeries Incident report.

  Problem bypass: Looking further into our environment file, we saw **LC_TIME=”C”.** Setting this value to blank will take care of the problem. The Domino development team continued to investigate the reason behind the way the system behaved.

  Late breaking news: Domino development wrote a problem report to document the above referenced problem. A fix was created and successfully tested. The description of the domino problem and its fix follow.

  ```
  Company: IBM Level 2 Large Systems Support  Incident: 994053

  Subject: Current Status - SPR RSCF45ULP2 opened. Fix piloted OK on
  Customer system.

  Note:

  Private Note:  The root of the problem seems to be that setlocale()
  function is returning a multi-value string to Lotus Script init on the
  customers system.  After further investigation, I found that the
  cckSTRConvPlatToCSID() function expects a string containing a single
  value (like "C" or "en_US") and uses it to do a table lookup for the
  csid.  The multivalue string doesn't compare to anything in its table,
  so it returns the x1F default csid instead of the x20 that we get on the
  test system.  This is what causes the registration problems with mqlsx.

  So if and when a multivalue string is returned from the setlocale() (in
  the GetLanguageSetting() function in cls.cpp), it has to be pruned to a
  single value. It appears other platform(s) return a string with values
  separated by a space, as there is already code in this area of cls.cpp
  to prune the string to the first value.  For OS/390, the multivalue
  string is comma delimited and similar code is needed.

  Opened spr RSCF45ULP2 to address this.  The spr is targeted for the next
  4.6.4 build.  I have made the change and tested it.  It is available for
  integration.
  ```

MQSeries-CICS Bridge

<u>Connecting to CICS</u>

As a prerequisite for running the MQSeries-CICS Bridge, we connected our MQSeries subsystem to our CICS subsystem by enabling the MQSeries-CICS adapter (see the sections on the MQSeries-CICS adapter in the *MQSeries for MVS/ESA System Management Guide*.)

Modifying the OS/390 CICS Application to Interface with MQSeries via the MQSeries-CICS Bridge

Our existing inventory control application running on OS/390 consists of 3270 terminal-driven CICS transactions, which run programs that make SQL calls to DB2 for database queries and updates. We chose one of these transactions to copy and modify so that it could talk via MQSeries to the application on our NT servers. Naturally, we chose a transaction which performs a function similar to that required by our application: Check to see that the required part number exists in the inventory database, and then decrement the quantity available. Consequently, we only had to be concerned with altering the transaction program to receive input and respond via MQSeries, and did not have to make any substantial changes to the logic of the program.

Possible Approaches

There are a variety of approaches available for enabling communication between MQSeries and an existing CICS transaction. No single approach is the best choice for every environment. A significant factor in making the choice is the set of characteristics of the existing transaction, which on our OS/390 test system was written in COBOL and expected 3270 input streams. In particular, the transaction expected a single screen of input and would return a single screen of output. This existing enterprise transaction was written in such a way that the end-user interface (3270 BMS maps) was intertwined with the actual business logic — in the original development, there was no attempt made to separate the "presentation tier" from the "business logic tier."

At the time of our investigation, several possibilities were available for connecting MQSeries to this type of CICS transaction. We considered:

- The MQSeries CICS Adapter shipped with MQSeries — this is the CKTI "trigger monitor" CICS transaction that starts a CICS transaction when an MQSeries event occurs, for example when a message is put onto a specific queue. Using this approach would require that we modify our CICS transaction to use the MQSeries APIs, or write a separate front-end application that would be kicked off by CKTI, be able to put/pull messages on/off MQ queues via the MQ APIs, and then invoke our transaction by some other means on the back end.
- The MQSeries-CICS Bridge product, which enables an MQ application — not running in a CICS environment — to run a program on CICS for MVS/ESA and get a response back. This non-CICS application runs from any environment (in this case, Windows NT) with access to an MQSeries network that includes MQSeries for MVS/ESA. The CICS program is invoked using the EXEC CICS LINK (EXCI) command, but does not use a CICS terminal.

  This approach doesn't require the transaction program to use or understand the MQSeries APIs, but it does require that the transaction be invoked via an interface (EXCI) that doesn't work (directly) with transactions requiring 3270 input/output like ours.
- A "roll your own" MQSeries application, which could be written to retrieve the MQ message from the queue, then invoke the CICS transaction directly, either through EXCI or some other means. In effect, this would be like writing our own MQSeries-CICS Bridge.

All of these possibilities would have required us to write some code; there was no way to directly invoke our existing transaction as it was currently coded, since it couldn't handle the EXCI interface. So we had to decide what sort of application modification and/or development approach to use. We considered two choices:

- An external presentation interface (EPI) or front-end programming interface (FEPI) could be used as a "wrapper" around our existing CICS transactions. EPI is multi-platform, while FEPI is exclusively targeted at CICS on OS/390. FEPI allows you to write a program that acts as an automatic operator, or front end, to 3270-based back-end applications. FEPI functions as a terminal to the back-end CICS transactions, simulating 3270 input/output to/for the transaction. FEPI could be used in conjunction with the above methods to provide "glue" to access our old transactions without our rewriting the transactions to be invoked via EXCI. For example, the MQSeries-CICS Bridge could be set up to link to a separate "glue" application, which would then use FEPI to call our back-end CICS transaction using the 3270 interface it expects. Or, a "roll your own" MQSeries application could be written to accept MQ messages on the front end and call our CICS transaction on the back end via FEPI.
- We could modify our existing CICS transaction to be invoked by EXCI, and then use it in conjunction with one of the above methods that uses that interface.

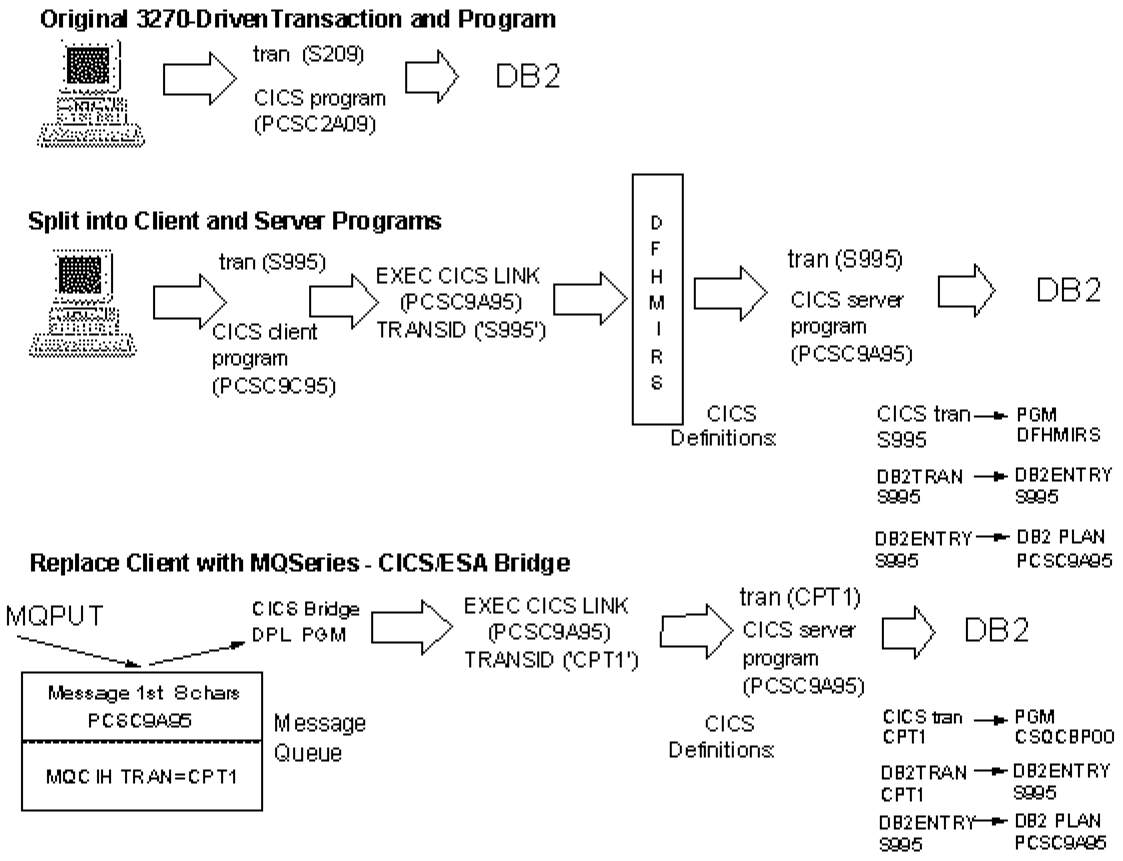
Chosen Approach:

We gave a lot of consideration to various approaches involving FEPI. However, we finally decided to modify the transaction to be invoked via EXCI, and then use the MQSeries-CICS Bridge. This choice was made for several reasons:

- Since our existing transactions used a fairly simple, single-screen approach for input/output, it was just as easy to modify the transaction as it would have been to write a FEPI wrapper. (For a transaction with multiple cascading input screens, it's not clear if this would also be the case.)
- By modifying the transaction we were able to separate presentation logic from business logic, giving us a more flexible transaction which could be used again easily for other future environments, accessed by a variety of front ends.
- By using the Bridge, we didn't have to further modify the transaction to handle MQSeries communication. The work of moving messages to/from the MQSeries queues was completely handled by the Bridge.

Migrating our existing 3270 transaction to use MQSeries:

As a first step, we modified the transaction by separating it into a pair of transactions: a front end (client) which would handle all screen I/O, and which would call (via EXCI) the back end (server) to handle all the business logic (DB2 SQL calls, etc.). Once those two transactions were finished and working, we swapped out the front-end transaction and replaced it with the MQSeries-CICS Bridge, which also used EXCI to invoke the server transaction. The figure below illustrates the phases of this migration:

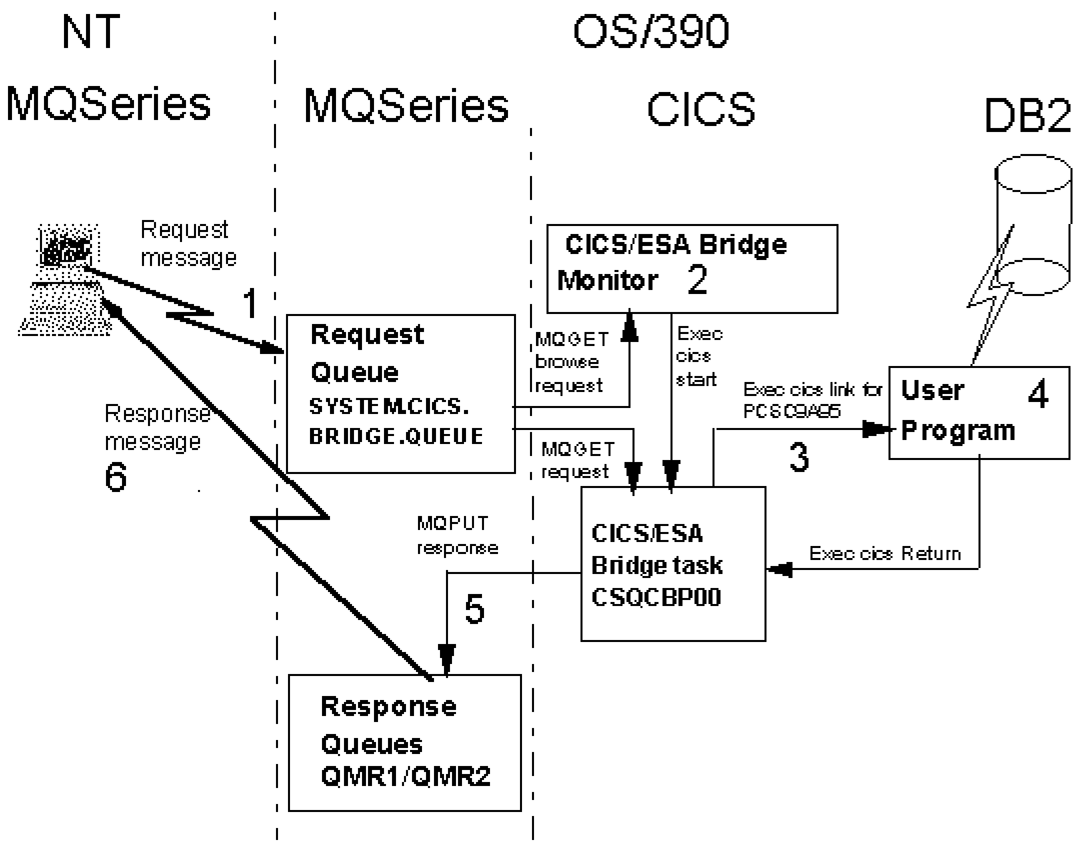

Original 3270-Driven Transaction and Program
tran (S209)
CICS program (PCSC2A09)
DB2
Split into Client and Server Programs
tran (S995)
CICS client program (PCSC9C95)
EXEC CICS LINK (PCSC9A95) TRANSID ('S995')
DFHMIRS
tran (S995)
CICS server program (PCSC9A95)
DB2
CICS Definitions:
CICS tran S995 → PGM DFHMIRS
DB2TRAN S995 → DB2ENTRY S995
DB2ENTRY S995 → DB2 PLAN PCSC9A95
Replace Client with MQSeries - CICS/ESA Bridge
MQPUT
CICS Bridge DPL PGM
Message 1st 8chars PCSC9A95
MQCIH TRAN=CPT1
Message Queue
EXEC CICS LINK (PCSC9A95) TRANSID ('CPT1')
tran (CPT1)
CICS server program (PCSC9A95)
DB2
CICS Definitions:
CICS tran CPT1 → PGM CSQCBP00
DB2TRAN CPT1 → DB2ENTRY S995
DB2ENTRY S995 → DB2 PLAN PCSC9A95

CICS Application Migration to MQSeries-CICS Bridge

As shown in figure above, these phases involved:

- Splitting the transaction program (PCSC2A09) into a client program (PCSC9C95) that does the 3270 I/O and a server program (PCSC9A95) that does the database access
- Using the EXCI command in the client program to call the server program
- Creating new TRANSACTION (CPT1) and DB2TRAN (CPT1) definitions so that the appropriate DB2 Plan would get invoked for our server program
- Creating a local MQ queue to be monitored by the MQSeries-CICS Bridge
- Creating a remote MQ queue corresponding to each MQ Series server, which the application program running on that server would identify as the "reply-to-queue" used to return a message

MQSeries-CICS-DB2 Message Flow:
This modification resulted in the envisioned message flow shown in the following figure and described below.



MQSeries Message Flow

- The Tier 2 application (AIX, AS/400, OS/2, Solaris or Windows NT) would put an MQ message on a queue, formatted to the specifications required by the MQSeries-CICS Bridge.

1. The Bridge's Monitor program would spot the message and issue an EXEC CICS START to the Bridge task (CSQCBP00).
2. The Bridge task would extract our "server" transaction program name and appropriate parameters, and invoke that transaction program via EXCI.
3. The server transaction program would run, update the DB2 tables as needed, and return a response to the Bridge task.
4. The Bridge task would format the response into an MQ message and put it on the appropriate response queue.
5. The MQ message would be returned to the Tier 2 application.

MQSeries-IMS Bridge

Making the IMS adapter available

In order to make the IMS adapter available we had to:

- Define MQSeries to IMS as a ESAF (external subsystem attach facility)
- Include the MQSeries load library MQS.V1R2M0.SCSQAUTH and MQS.V1R2M0.SCSQANLE in the JOBLIB or STEPLIB in your IMS Proc JCL
- Add CPIT and MQ test teams to Subsystems CSQP/MQMP and CSQ1/MQM1 in CSQQDEFV
- Include updated module CSQQDEFV in a user library in the CTRL region and Dependent region concatenations.
- Create a language interface module.
- Ensure that MQSeries has authority to issue MODIFY command.

Customizing the MQSeries-IMS Bridge

- Define the XCF and OTMA parameters for MQSeries
- Define the XCF and OTMA parameters to IMS

Get IMS Bridge MQPut sample program

- down load the sample MQPUT program called MA1C from http://www.software.ibm.com/ts/mqseries/txppacs/txpm3.html

Pick an IMS Transaction

We chose IT8 CPIT workload transaction to test with for simplicity and ease of use. The IT8 transaction will update an IMS table with caller parameters. The parameters were sent via input file to the MQPUT program. The IT8 transaction will send a reply back to the caller to inform with either successful updates or a failure.

Testing the IMS Bridge

- Start the MQ MGR, Channel Initiator, and Listener.
- Recycle IMS using the new updated IMSMQ proc, start up MPRCPIT0.
- Put a message on the MQSeries-IMS Bridge
  The message will contain the IT8 IMS transaction and the passed input file for the MA1C program. Following is a sample of the MA1C program input file.

```
TESTSTART
DEBUGLEVEL         1
NOOFMESSAGES       1
QMGRNAME          CSQP
QNAME             SYSTEM.IMS.BRIDGE.QUEUE
OPENOPTIONS        16
MQMDREPORT         0
MQMDMSGTYPE        1
MQMDFEEDBACK       0
MQMDENCODING       785
MQMDFORMAT         MQIMS
```

```
MQMDPRIORITY          1
MQMDPERSISTENCE    1
IMSFORMAT                 MQIMS
IMSMFSMAPNAME       IT8MID
IMSREPLYTOFORMAT  MQIMS
IMSLL                       104
IMSZZ                       0
TESTEND
```

We were able to put a message on the MQSeries-IMS Bridge and kick off the IT8 transaction. The IMS transaction returned a successful reply back to caller.

The following is the output received on the MQ replyto queue.

```
MQOPEN - 'CSQPMQQ'
MQGET of message number 1
****Message descriptor****
0  StrucId  : 'MD  '  Version : 1
   Report    : 0  MsgType : 2
   Expiry     : -1  Feedback : 0
   Encoding : 785  CodedCharSetId : 500
   Format    : 'MQIMS   '
   Priority    : 1  Persistence : 1
   MsgId     : X'C3E2D840C3E2D8D74040404040404040B0D0900875B2D081'
   CorrelId  : X'C3E2D840C3E2D8D74040404040404040B0D09007BF936480'
   BackoutCount    : 0
   ReplyToQ          : '                                                '
   ReplyToQMgr     : 'CSQP                                            '
** Identity Context
   UserIdentifier       : 'RAMSES      '
   AccountingToken :
   X'13C4F7F1F1F5D76BC4F1F06BC2F9F2F2F2F1F4F100000000000000000000000000'
   ApplIdentityData  : '                                '
** Origin Context
       PutApplType    : '20'
      PutApplName    : 'IMSPET  IMSPETZ0            '
      PutDate            : '19980727'    PutTime  : '19373853'
      ApplOriginData : '    '
****   Message      ****
 length - 148 bytes

 00000000:  C9C9 C840 0000 0001 0000 0054 0000 0311             'IIH ............'
 00000010:  0000 0000 D4D8 C9D4 E200 0000 0000 0000             '....MQIMS.......'
 00000020:  4040 4040 4040 4040 C9E3 F8D4 C9C4 4040             '        IT8MID  '
 00000030:  4040 4040 4040 4040 4040 4040 4040 4040             '                '
 00000040:  0000 0000 0000 0000 0000 0000 0000 0000             '................'
 00000050:  40F0 4040 0040 0300 C1D3 D340 D6C6 40E3          ' 0  . ..ALL OF T'
 00000060:  C8C5 40D7 C5D9 E3C9 D5C5 D5E3 40C4 C1E3          'HE PERTINENT DAT'

 00000070:  C140 C2C1 E2C5 E240 C8C1 E5C5 40C2 C5C5          'A BASES HAVE BEE'

 00000080:  D540 C3C1 D9C5 C6E4 D3D3 E840 E4D7 C4C1          'N CAREFULLY UPDA'
 00000090:  E3C5 C43F                                        'TED.           '
 1

No more messages
MQCLOSE
MQDISC
```

Experiences with the AIX platform

MQSeries link LotusScript Extension (MQLSX)

Installing

The product documentation addresses installing MQLSX from CDROM. Here is a method of getting images from the Web and installing them:

1. Get MQLSX for AIX from the Web.
    a. http://www.software.ibm.com/ts/mqseries/txppacs/txpm2.html
    b. Choose MA6D "MQSeries for AIX link LotusScript Extension"
    c. Note: MA05 only points to MA6D
    d. Download ma6d.tar.Z from the Web page
    e. Copy it to a subdirectory
    f. uncompress ma6d.tar.Z
    g. tar -xvf ma6d.tar
    h. (as root) inutoc .
    i. smitty install
        1. use that dir as the install directory
        2. choose mqlsx (all) - it's the only one...
2. There is also a User's Guide in PDF format available on the Web page.

Hints & Tips

- AIX conversions needed

  We had a MQSeries C program that could talk to the S/390 in Poughkeepsie with no problems. The Channel Definition on AIX had CONVERT(YES) so the ascii-ebcdic conversion was handled by MQSeries.

  We also had a MQLSX Notes database that did the same thing to POK. It was failing on the PUT with Reason Code=2119 which is NOT_CONVERTED.

  MQLSX must handle the conversion. We had to
  ```
  export GMQ_XLAT_PATH=/usr/lpp/mqm/mqlsx/conv
  ```
  prior to starting Notes. If this environment variable is not set, then no conversion is possible (thus the 2119 Reason Code). This environment variable is not automatically set by installation.
- AIX authorizations required when defining MQ Series objects.

  The MQLSX sample database (connect to POK) got an error. The log file said:
  ```
  Reason Code:2035
  ```
  Looking in /usr/lpp/mqm/inc/cmqc.h
  This is: #define MQRC_NOT_AUTHORIZED 2035L

  It turned out that we defined the Q's using userid=mqm
  Looking at that userid (smitty users -> List Characteristics), we found:
  ```
  Group SET          [mqm,staff]
  ```

Looking at userid=notes (where we ran the sample application), we found:

```
Group SET        [notes,dba]
```

To make the userid=notes able to access the Q's defined by userid=mqm, we needed to make them part of the same Group SET. We used smitty again to add "mqm" to the notes Group SET.

```
Group SET        [notes,dba,mqm]
```

Note: You must exit completely and log back in as userid=notes to make this take effect.

- AIX results from extended run
  The AIX MQLSX application was set up for a longer run to see how well it ran. We defined the transaction to be:
  a. send a part number to POK via a MQLSX application in a Domino database
  b. the network consists of multiple hops and T-1 lines crossing state lines and multiple regional Bell Co.'s
  c. the AIX machine was dedicated to this test
  d. the 390 in POK had an unknown load
  e. the 390 received the request (via MQ) and sent it over the MQSeries-CICS Bridge to a CICS transaction.
  f. the CICS transaction made the request and shipped a single value back to us (quantity on hand)

  With all that said, the best we did was 24000 transactions in 61 minutes.

Experiences with the AS/400 platform

MQSeries link LotusScript Extension (MQLSX)

Overview

For the AS/400 you must use the readme.txt for install and general MQLSX directions. The sample QMQLSAMP.NSF shipped with and discussed in the MQLSX Users Guide is not intended to work with the AS/400. Due to the nature of the OS/400 Operating System, it is important that a user of MQLSX with an AS/400 understand the following items.

- All activities must be run as an MQLSX Agent , and to edit the agents from a Notes Client, the MQLSX Client software must be installed on it.
- When you run an agent it must be scheduled so it will be run from the Domino server, i.e., you can not just choose run for the agent from the Notes Client.

Additional information can be found at http://www.software.ibm.com/ts/mqseries/.

Installation of the MQLSX product "MQSeries for AS/400 link LotusScript Extension"

The current product at this time is 5733MQX, and can be downloaded at the following location http://www.software.ibm.com/ts/mqseries/txppacs/ma43.html. Follow the readme.txt file included regarding the AS/400 install, refer to additional FTP directions below if needed. We used the MQLSX Client product for Windows NT. It can be downloaded from the Web at - http://www.software.ibm.com/ts/mqseries/txppacs/ma7d.html MA7D: MQSeries for Windows 32-bit platforms link LotusScript Extension.

Specific MQLSX programming considerations and suggestions

- Scheduling - The frequency of scheduling of agents can be set to a minimum of 30 minutes, or you can force the agent to run immediately by selecting the check mark to turn the schedule on and off. This is useful to see if changes made to the agent work as desired.
- Troubleshooting - We found it useful to use all of the following error sources.
  - Domino Console - (DSPDOMCSL "servername", or DSPDOMCSL "servername". When the agent is run you will see general Notes errors and by inserting Print statements in the Agent you can display variable values or check completion codes such as. 'Print "mqmsg.ReplyToQueueName:" & mqmsg.ReplyToQueueName, Print "mqmsg.CompletionCode" & mqmsg.CompletionCode.
  - Trace file - You can create trace files to include Variable Values, Completion Codes and pass error messages. Samples like TraceString tag & " Completion Code: " & v.completioncode, TraceString tag & " ReasonCode: " & v.reasoncode.
  - Run the Agent - Action/Run. Although this will fail on all MQSeries calls with the AS/400 — because it is running the Agent on the local machine rather than at the server — it may be useful and a fast way to track the calling of other subs or functions within your Notes Database.
  - Use Documents - Pass variables into and results out for Puts and Reads.
  - Print out the error files generated by MQLSX - The error will be reported but not the definition. If MQSeries is installed on AS/400, this header file is also installed in the QMQM library, H file, CMQC member. Samples #define MQRC_NOT_AUTHORIZED 2035L, #define MQRC_STORAGE_NOT_AVAILABLE 2071L

- For the 2035 error mentioned above - It is very important the Authority be granted to the MQ objects on the AS/400 for user QNOTES. As all MQSeries names will be unique per site this is done per MQ Object used. Sample - GRTMQMAUT OBJ(SYSTEM.DEFAULT.LOCAL.QUEUE) OBJTYPE(*Q) USER(QNOTES) AUT(*READ *ADD *DLT *OBJOPR)

- Items to check on the Notes Database that will execute MQLSX agents.
  - Add the Domino server as Server/Manager to the ACL for the DB.
  - Add or change in the ACL the Administration Server to be the server that you expect to run the MQLSX agents.
  - Check the Name and Address book Server/Server as to who can run agents, add Notes users or groups who will start and stop the MQLSX agents.

<u>Hints & Tips</u>

- FTP directions - to an AS/400 from a PC for the MQLSX product
  - cd to the sub directory where the downloaded file "savefile" is stored.
  - FTP directions - to AS/400 from a PC.
  - Open Dos Window
  - ftp "system_name"
  - prompted for signon
  - prompted for password
  - bin " to put in binary format "
  - cd /qsys.lib " enter into the lib area of AS/400 " " this will also set your name format to 1"
  - cd /MA43.lib " the lib that the readme.txt file instructed to create "
  - put savefile savefile.savf
  - quit

- To install the MQLSX SupportPac: README.TXT
  - Create a temporary library called MA43 on your AS/400:
    CRTLIB LIB(MA43)
  - Create a savefile in library MA43 ready to receive the SupportPac:
    CRTSAVF FILE(MA43/SAVEFILE) TEXT('MA43 SupportPac savefile')
  - Set the default receiving data library by making MA43 the current AS/400 library:
    CHGCURLIB CURLIB(MA43)
  - Transfer (in BINARY) the SAVEFILE shipped in the SupportPac from your workstation to replace the object SAVEFILE in library MA43.
  - Install the MQLSX licensed program on your AS/400 from the savefile:
    RSTLICPGM LICPGM(5733MQX) DEV(*SAVF) SAVF(MA43/SAVEFILE)
  - The RSTLICPGM command will install the product documentation into the directory /QIBM/ProdData/MQLSX/docs. The documentation is available as a Lotus Notes database, an Adobe Acrobat Reader PDF file, and as a PostScript file. If you do not have an Adobe Acrobat Reader on your workstation, you can obtain one (free) via:
    http://www.adobe.com/prodindex/acrobat/readstep.html

- Setting the GMQ_XLAT_PATH and GMQ_TRACE_PATH environment variables is not necessary, except to override the provided defaults:
  GMQ_XLAT_PATH = /QIBM/ProdData/MQLSX/conv
  GMQ_TRACE_PATH = /QIBM/UserData/MQLSX
  To set other environment variables (GMQ_TRACE and GMQ_TRACE_LEVEL) so that they will affect the Domino server jobs where your LotusScript programs will be running, you will need to do the following:
  a. Use ENDDOMSVR to terminate the Domino server
  b. Use ADDENVVAR to set the desired environment variables (use WRKENVVAR to view your environment variables)
  c. Use STRDOMSVR to restart the Domino server; your environment variables will be propagated to the Domino server jobs
- In order to write LotusScript programs for use with MQLSX for AS/400, you will need to have MQLSX installed on the Notes client where the LotusScript programs will be edited. This is because LotusScript programs for Domino are compiled on the Notes client where the program is being edited. MQSeries is not required to be installed on the Notes client for the compilation of agents, just the MQLSX itself. An MQSeries Client or Server would be required on the Notes client in order to execute the LotusScript locally. See the MQLSX User Guide for MQSeries requirements.
- Getting Help
  - The MQSeries Web site is located at: http://www.software.ibm.com/ts/mqseries/
  - The SupportPac page is: http://www.software.ibm.com/ts/mqseries/txppacs
  - For current information on known problems and available fixes please see the Support option on the MQSeries Web site at:
    http://www.software.ibm.com/ts/mqseries/support

Late-Breaking News

- Lotus Notes Release
  This release of MQLSX has been tested against Release 4.6 of Lotus Domino Server for AS/400.
- MQSeries Release
  This release of MQLSX has been tested against V4R2M0 of MQSeries for AS/400.
- Syncpoint and Commitment Control
  MQSeries for AS/400 does not implement the MQCMIT and MQBACK functions. MQLSX uses the AS/400's commitment control facilities to implement syncpoint control of MQSeries messages, as described in the MQSeries Application Programming Guide. This is transparent to the LotusScript program; the use of MQPMO_SYNCPOINT in the Options property of the MQPutMessageOptions class, the use of MQGMO_SYNCPOINT in the Options property of the MQGetMessageOptions class, and the use of the Commit and Backout methods of the MQQueueManager class are exactly the same as on any other MQLSX platform.
- Using MQLSX from Domino Web agents with multi-threading
  If MQLSX will be used in Domino Web agents, then DominoAsynchronizeAgents must not be set to 1 in the notes.ini file. MQSeries for AS/400 and the AS/400's commitment control facilities are not thread-safe, and setting DominoAsynchronizeAgents to 1 could result in

multiple Web agents using MQLSX running simultaneously in different threads of the HTTP server.

This restriction only applies to the Domino servers where the MQLSX Web agents will be run. If you have several partitioned Domino servers on your AS/400, you can set DominoAsynchronizeAgents to 1 on the other Domino servers if they will not be running MQLSX Web agents. This gives much improved performance for web-based applications.

- Dynamic loading of MQSeries service program
  The service programs for MQSeries are dynamically loaded when your application calls the MQLSX. If you have a problem, (say you see the reason code MQRC_LIBRARY_LOAD_ERROR), please refer to the information that follows.

  When the first call is made to the MQI, using one of the MQLSX methods, MQSeries support is dynamically loaded by the MQLSX. For example, the AccessQueueManager method issues an MQCONN call that the MQLSX dynamically resolves.

  On AS/400 systems, the MQSeries support will be obtained from the MQSeries server using QMQM/AMQZSTUB *SRVPGM. The MQLSX requires access to this object for the dynamic load to succeed.

  If the MQSeries objects cannot be found, your program will fail with Return Code 6000 (MQRC_LIBRARY_LOAD_ERROR).

  What you can do:
  a. Ensure that QMQM/AMQZSTUB *SRVPGM exists. If not, reinstall MQSeries for AS/400.
  b. Ensure that *PUBLIC has *USE authority to QMQM/AMQZSTUB *SRVPGM.

- MQSeries link sample
  If you run the MQLSX Link sample application, you may need to change the Agent settings. In particular:
  AMgr_DocUpdateAgentMinInterval=1 (default is 30)
  AMgr_DocUpdateEventDelay=1 (default is 5)
  Setting these to 1 will cause the Agent to run as soon as it can. However, you should set them according to the workload on your Server. See 'About the NOTES.INI file' in the Notes Administration Guide for more information.
- International Character Set Support
  MQLSX provides support for a wide variety of different character sets. For more information please refer to the readme.ccs file in the MQLSX conv sub-directory which lists the conversions supported. Requests for the support of additional and future character sets should be addressed via your IBM representative.
- Trace level with multi-threading
  The GMQ_TRACE_LEVEL environment variable controls the level of detail you want recorded in your trace file (see MQLSX User's Guide Chapter 4). When running

multi-threaded the numeric value component of GMQ_TRACE_LEVEL only controls the first 40 threads used. Subsequent threads are traced at level 9.

<u>MQSeries for the AS/400 experiences</u>

- It is always recommended to check for current product PTFs. Make sure that you check the cover letters before applying.
- Installation problems seen
  - After reinstalling MQ product you need to create a new Queue Manager - if this step does not complete - appears to hang the session after issuing the CRTMQM command. These are nearly always caused by the existing QMQM activation group within your job. If you delete and reinstall the product, after using MQSeries in your job, it is essential to reclaim all eligible activation groups within your job or you may end up trying to run lost objects (pointers to unknown locations etc.) .

    i.e. - sample scenario
    Delete Licensed Program
    RCLACTGRP *ELIGIBLE
    Install Licensed Program
    CRTMQM NEWQMGR
    RCLACTGRP *ELIGIBLE
    CALL QMQM/AMQSDEF4

    If in doubt, sign off and sign back on again before creating your new Queue Manager. This problem should be fully addressed with the next release of MQSeries for the AS/400.
- Problems running the sample C PUT program distributed with the AS/400 MQSeries, AMQSPUT4, AMQSGET4
  - You need to link a service program when you compile the sample program - found in MQSeries Application Programmers Guide. In MQSeries for AS/400, you must bind your ILE C/400 programs and RPG/400 static calls to the supplied AMQZSTUB service program. When using the AMQSPUT4 and AMQSGET4 sample applications on the AS/400, the following changes need to be noted:
    - Change the line in the program which looks like this: #include <cmqc.h> /* includes for MQI */ ->change it to: #include "cmqc.h"
    - Change the lines in the program from:
      int main(argc, argv)
      int argc
      char **argv
      Change to:
      int main(int argc, char **argv)
    - Successful compile of the AMQSPUT0 sample c program on 400 CRTCMOD MODULE(QMTEST/TEST) SRCFILE(QMTEST/QCSRC) SRCMBR(AMQSPUT0) TEXT('TEST COMPILE PUT'). Then create a program - binding to AMQZSTUB - service program CRTPGM PGM(QMTEST/AMQSPUT0) MODULE(QMTEST/TEST) TEXT('Get program for CICS bridge test') BNDSRVPGM(QMQM/AMQZSTUB) Program AMQSPUT0 created in library QMTEST.

- The program is designed to take input data from a file - You pass the file name at the time you call the program - see sample. CALL PGM(AMQSPUT0) PARM('QMQMSAMP/AMQSDATA(PUT)'). The Sample program must include the queue writing to and Data - No mention of the QMGR.

- For beginning users of MQSeries on the AS/400 - Procedures to follow when initially setting up MQSeries for AS/400:
  1. Install the current MQSeries product (5769MQ2 - Base and options), and apply the latest PTFs
     - RSTLICPGM (to delete the product, use the command DLTLICPGM)
  2. At an AS/400 command line, type GO CMDMQM to view a menu of the various MQSeries commands
  3. ADDLIBLE QMQM
  4. CRTMQM MQMNAME(qmgrname) UDLMSGQ(SYSTEM.DEAD.LETTER.QUEUE) DFTTMQ(xmitqname)
     - This command creates the MQ queue manager
  5. STRMQM MQMNAME(qmgrname)
     - This command starts the MQ queue manager
  6. STRPDM, option 3: Work with the CL program (AMQSDEF4) in file QCLSRC in library QMQMSAMP. Compile this program and call it to set up the default MQSeries queues/channels.
     - CALL PGM(QMQMSAMP/AMQSDEF4)
  7. Look at the default MQSeries queues and channels, type GO CMDMQM:
     - Option 19 (Work with MQM Queues) or WRKMQMQ command
     - Option 41 (Work with MQM Channels) or WRKMQMCHL command
  8. Because this is the first time doing some of this MQ set up, and it may be necessary to either delete the Queue Manager or the MQSeries product, it is nice to have a CL program that can easily recreate your environment specific local/remote queues, sender/receiver channels, etc. Do the following:
     - Create a test library -> CRTLIB(QMTEST)
     - Create source physical file of QCLSRC in QMTEST library -> CRTSRCPF FILE(QMTEST/QCLSRC)
     - Copy the AMQSDEF4 source in the QCLSRC file in library QMQMSAMP to 'newfilename' in QCLSRC in QMTEST
     - Edit the 'newfilename' file using -> WRKMBRPDM FILE(QMTEST/QCLSRC) and 'option 2' to edit the file. Input your specific local/remote queue information, sender/receiver channel information, transmission queue information, etc. Then compile the CL Program, using 'option 14'.
     - At a command line type -> CALL PGM(QMTEST/newfilename), to set up the queues/channels/etc.
     - Make sure the Queue Manager is running -> GO CMDMQM, option 6.
  9. Next start the Listener and Channels
     - Start the Listener -> STRMQMLSR
     - Start the Sender and Receiver Channels -> STRMQMCHL or Start the MQM Channel Initiator - > STRMQMCHLI (to use this second option, the transmission queue needs to be associated with a channel initiation queue). This can be set up when you create

the transmission queue -> CRTMQMQ QNAME(XMITQ) INITQNAME(SYSTEM.CHANNEL.INITQ), and this can be done as part of the automated CL Program used in section 8 above.

- If the other system that you are communicating with is an AS/400, repeat steps 1-9 on the other AS/400. If it is another platform follow the steps documented for installation and set up of MQSeries on that platform.
- Check MQM Authority - Option 26 from GO CMDMQM - or DSPMQMAUT, Add additional users as needed and QNOTES user if using MQLSX product. Only the creator of the object has all authority as does system supplied QMQM user.
  - GRTMQMAUT OBJ(SYSTEM.DEFAULT.LOCAL.QUEUE) OBJTYPE(*Q) USER(QNOTES) AUT(*READ *ADD *DLT *OBJOPR

```
User          Authority  Opr  Mgt  Exist  Read  Add  Update Delete
*PUBLIC       USER DEF                      X
QMQM          *ALL        X    X     X      X    X      X     X
CREATOR ID    *ALL        X    X     X      X    X      X     X
QNOTES        USER DEF    X    X     X      X    X      X     X
```

- Once the systems are installed and configured for MQSeries, try running the MQSeries AMQSGET4 and AMQSPUT4 sample applications to a local queue to test your setup and to a remote queue to test the communications.

Experiences with the OS/2 platform

MQSeries link LotusScript Extension (MQLSX)

Installation

Our installation of MQSeries on OS/2 went very smoothly. No problems were encountered. The installation procedures are documented in *MQSeries for OS/2 Warp V5.0 Quick Beginnings.*

To complete the process, we also installed:

- MQSeries for OS/2 Version 5.0 + CSD level U200069
- MQEI 1.0 + CSD U200085
- MQLSX 1.3.2

The most recent CSD can be located at the MQSeries Support - Fixes Web page
http://www.software.ibm.com/ts/mqseries/support/summary/os2.html.

Initial verification (using Windows NT-to-OS/2 -MQSeries Connection as an example)

For testing of our Windows NT to OS/2 MQSeries connection, we performed the following on the Windows NT side. Note that each command was issued from its own DOS window.

Issued command **STRMQM QMO2**
Issued command **RUNMQLSR -t TCP -m QMO2**
Issued command **RUNMQCHI**
Issued **RUNMQSC** from a MS-DOS window

a. from this prompt issued **START CHANNEL(O2TOSBRT)**
b. displayed the channel to OS/2 by issuing: **DISPLAY CHSTATUS(O2TOSBRT)** which should show running if the channel is active
c. displayed the channel from OS/2 by issuing: **DISPLAY CHSTATUS(OSBRTO2T)** which should show active if the channel has been activated in OS/2

Note: Steps a-c from the next section must also be run on the remote server before the channels will show up as active.

For the testing on the OS/2 side, we ensured that TCP/IP was active and did the following:

Issued command **STRMQM OSBR**
Issued command **RUNMQLSR -t TCP -m OSBR**
Issued command **RUNMQCHI**
Issued **RUNMQSC** from a MS-DOS window

a. from this prompt issued **START CHANNEL(OSBRTO2T)**
b. displayed the channel to NT by issuing: **DISPLAY CHSTATUS(OSBRTO2T)** which should show running if the channel is active
c. displayed the channel from NT by issuing: **DISPLAY CHSTATUS(O2TOSBRT)** which should show active if the channel has been activated in NT

Creating and Starting qmgr, queues, channels

Once the above installation was complete, we created the necessary local queues, remote queues and channels for linking with MQSeries on NT in Austin, Solaris in Dallas, and AIX in Austin. We used our OS/2 Queue Manager in our setup, OSBR, which was used in conjunction with the CPIT MQSeries-CICS Bridge on OS/2. The commands below were used to create and define Queue Manager OSBR.

To complete the setup for OSBR, issue the same commands listed below.

- Create Queue manager OSBR, with dead letter queue OS.DLQ

  CRTMQM -q -u OS.DLQ OSBR

  Where, **-q** is for creating a default queue manager (required for the bridge) and **-u** is for creating the dead-letter queue, **OS.DLQ**, for qmgr names **OSBR**
- Start Queue Manager **OSBR**

  STRMQM OSBR
- Create the base definitions for connecting to remote Queue Managers.

  RUNMQSC OSBR < MQHOSTS.ALL

  Note: This should complete with all "commands read", no "syntax errors" and no commands "not processed." If there are any problems here, reissue the command as follows:

  RUNMQSC OSBR < MQHOSTS.ALL > ERROR.OUT

  Then use any text editor to check the **ERROR.OUT** file for what the problems are.

  The definitions that we used for the channels and queues are available in Appendix B - MQSeries Object Definitions.

- We defined the following MQ objects to enable communications between the Queue Manager on OS/2 in Austin and the Queue Managers in remote locations:
  - Local message queue (defined to remote Queue Manager as a remote queue)
  - Remote message queue (defined to remote Queue Manager as a local queue)(Note 1)
  - Transmission queue for sending messages to remote Queue Manager through Sender Channel
  - Sender channel for sending messages to remote Queue Manager — this definition contains the IP address for remote Queue Manager, IP port, and the Transmission queue name
  - Receiver channel for receiving messages from remote Queue Manager

  Note 1: The Remote queue must not be defined for the MQEI connection on the Tier 3 OS/2 bridge side.

Hints and Tips

- When creating a Queue Manager (qmgr), keep these things in mind:
  - Each node should have a default qmgr.
  - If you run the runmqsc command without specifying your qmgr's name, your default qmgr will process the command and requests. Applications will also connect to it when you do not specify qmgr's name explicitly.
  - Specifying a default qmgr is optional, but strongly recommended.
  - There should be only one mqs.ini file on a given system. The system path statement should point to that.

    - Specifying a qmgr as the default replaces any existing qmgr specifications for the node.
    - If you do not have or you want to change the default qmgr, you can modify mqs.ini file by adding or modifying the stanza:
      ```
      Default QueueManager:
      Name=OSBR
      ```
      where, OSBR is the qmgr you had created without -q switch for default qmgr.

      Note: the stanza gets updated automatically when you create a new qmgr with '-q' switch. In this case, the last qmgr you create becomes the default qmgr. There should be only one default qmgr defined in the mqs.ini.
    - For a default qmgr, IP port (1414) is assigned by default by MQSeries. The remote qmgr needs not to assign a port for your local default queue manager unless you explicitly assign another port by modifying its mqs.ini file.
    - If the remote qmgr has assigned port (1414) in its Sender Channel definition for your default qmgr, and then you change your local default queue manager, or create a new one, then that remote qmgr will have connection problem. Assign a new port to that qmgr. The remote qmgr will have to make the appropriate change accordingly.
    - Do not add qmgr IP port to your services file (NT and OS/2).
    - Only the default qmgr should be using the port 1414.
- MQ Series qmgr does not start if you have installed IBM Visual Age C++. The OS/2 path statement in the config.sys file for DDE4MTH.DLL causes an access violation error. If you had previously installed MQSeries client in the same system, you will have to remove it along with its path statements from the OS/2 config.sys file.

Experiences with the Windows NT Platform

MQSeries Overview

MQSeries was used on Windows NT in two capacities.

1. As a Tier 3 system: Applications were executed from Tier 2 systems to access the database in Microsoft SQL Server on Tier 3.
2. As a Tier 2 system: MQLSX and MQEI applications were executed on this system to multiple Tier 3 systems.

<u>Installing</u>
Our installation of MQSeries on Windows NT went very smoothly. No problems were encountered. The installation procedures are documented in *MQSeries for Windows NT V5.0 Quick Beginnings*.

To complete the process, we also installed:

- MQSeries for Windows NT Version 5.0 + CSD number 01, corresponding to PTF U200068
- MQEI 1.0 + CSD 01
- MQLSX 1.3.2

The most recent CSD can be located at the MQSeries Support - Fixes Web page
http://www.software.ibm.com/ts/mqseries/support/summary/wnt.html.

Tier 3 system

<u>Creating and starting qmgr, queues, channels</u>
Once installation and configuration were complete, we created the necessary local queues, remote queues and channels for linking with MQSeries on AIX, OS/2 and Windows NT in Austin and Solaris in Dallas. The NT Queue Manager used in our setup, NTBR, was used in conjunction with CPIT MQSeries-CICS Bridge on Windows NT. The commands below were used to create and define Queue Manager NTBR.

To complete the setup for NTBR, issue the same commands listed below.

Create Queue manager NTBR, with dead letter queue **NT.DLQ**

```
CRTMQM -q -u NT.DLQ NTBR
```

Where, **-q** is for creating a default queue manager (required for the bridge) and **-u** is for creating the dead-letter queue, **NT.DLQ**, for QMgr name **NTBR**

Start Queue Manager **NTBR**

```
STRMQM NTBR
```

Create the base definitions for connecting to remote Queue Managers.

RUNMQSC NTBR < MQNTBR.TXT

Note: This should complete with all "commands read", no "syntax errors" and no commands "not processed." If there are any problems here, reissue the command as follows:

RUNMQSC NTBR < MQNTBR.TXT > NTERROR.OUT

Then use any text editor to check the NTERROR.OUT file for the problems.

Sample definitions for channels and queues are available in Appendix B - MQSeries Object Definitions.

We defined the following MQ objects to enable communications between the Queue Manager on Windows NT in Dallas and the Queue Managers in remote locations:

- Local message queue (defined to remote Queue Manager as a remote queue)
- Remote message queue (defined to remote Queue Manager as a local queue)(Note 1)
- Transmission queue for sending messages to remote Queue Manager through Sender Channel
- Sender channel for sending messages to remote Queue Manager — this definition contains the IP address for remote Queue Manager, IP port, and the Transmission queue name
- Receiver channel for receiving messages from remote Queue Manager

Note 1: Remote queue must not be defined for MQEI connection on the Tier 3 Windows NT bridge system.

To complete the preparation for linking to a remote queue manager, the following was done.

- Started the Listener with command, **RUNMQLSR -T TCP**
- Started the Channel Initiator, **RUNMQCHI**
- Started the Channel with command, **RUNMQCHL -M QMD1 -C NTBRTD1T**
- Started **RUNMQSC** to check status of channels and other objects
  From the prompt, we displayed the Sender status by issuing the command,

  DISPLAY CHSTATUS(NTBRTD1T)

  If the channel was running successfully, then **"STATUS(RUNNING)"** would be shown.

  Note: Once the Receiver channel has started, its status can also be shown in **RUNMQSC** by issuing the following **DISPLAY CHSTATUS(D1TNTBRT)**. If the channel was running successfully, then **"STATUS(RUNNING)"** would be shown.

<u>Hints and Tips</u>

When creating a Queue Manager (qmgr), keep these things in mind:

- Each node should have a default qmgr.
- If you run the runmqsc command without specifying your qmgr's name, your default qmgr will process the command and requests. Applications will also connect to it when you do not specify qmgr's name explicitly.
- Specifying a default qmgr is optional, but strongly recommended.

- There should be only one mqs.ini file on a given system. The system path statement should point to that.
- Specifying a qmgr as the default replaces any existing qmgr specifications for the node.
- If you want to setup a default qmgr after initial installation or if you want to change current default qmgr, you must modify the mqs.ini file by adding or modifying the stanza:

  ```
  Default QueueManager:
  Name=NTBR
  ```

  where, NTBR is the qmgr you had created without -q switch for default qmgr.

  Note: the stanza gets updated automatically when you create a new qmgr with -q switch. In this case, the last qmgr you create becomes the default qmgr. There should be only one default qmgr defined in the mqs.ini.
- For a default qmgr, IP port (1414) is assigned by default by MQSeries. The remote qmgr need not assign a port for your local default queue manager unless you explicitly assign another port by modifying its mqs.ini file.
- If the Tier 2 remote qmgr has assigned port (1414) in its Sender Channel definition for the Tier 3 default qmgr and, subsequently, the port of the local default queue manager is changed or created with a different port, then the Tier 2 remote qmgr will have connection problems. The connection problem can be resolved by assigning the correct port to the qmgr on the Tier 2 system.
- Do not add qmgr IP port to your services file (NT and OS/2).
- Only the default qmgr should be using the port 1414.

Configuring CICS for NT 4.0.0 for use with Microsoft SQL Server 6.5

At the time of this report, the CICS Administration Guide is somewhat unclear and can cause a lot of unnecessary work to implement this particular CICS-RDBMS connection. The information in this section is provided as a quick-fix to enable you to get this going as smoothly and quickly as possible. The part of the CICS Administration Guide pertaining to the configuring of the XAD Product definition is accurate and is not in question here. It is the SQL database source code examples that require clarification and that is what this section is providing.

In order to allow CICS to access SQL Server (and any other RDBMS) the database must supply an X/Open XA compliant interface (see X/Open Distributed Transaction Processing Reference Model ISBN 1 872630 16 2). The XA interface is not an ordinary API, but a system level interface between a transaction manager (TM) and resource manager (RM) in the X/Open DTP model. However there is some programming that needs to take place. The sample code included in Appendix D is written in C. This supposedly can be done in COBOL also, but we have not tried it. You must use Embedded SQL. We tried to use DBLIB (the call level interface for SQL Server) and ran into some pesky problems with that route so our advice is, "Don't try it!". The main problem encountered was this: When you set up an XAD product definition in your region, CICS establishes a long-running connection to the database. Any transactions in the region that need to access SQL Server will use that pre-established connection (or connections - one per server). Now, when that connection is established, it uses an arbitrary global name (supplied by you in your XAD Resource Manager Initialization String) so that any transaction that is loaded into that region can have access to that name. When you use Embedded SQL you can set your database connection to a previously established connection (EXEC SQL SET CONNECTION TO :connection) to access it. Don't worry about opening and closing database connections in your transactions, just referencing already established connections. The Resource Manager takes care of all the dirty work! We found that when using DBLIB, getting a handle to that pre-established connection was messy and buggy — again, you must use Embedded SQL!

What you will need:

- IBM Transaction Server 4.0.x and all the latest PTFs.
- Microsoft SQL Server 6.5 Developer's Resource Kit. (http://www.microsoft.com/sql/) Microsoft Embedded SQL for C (the only place we could find this was on the MSDN April 1998 Additional tools and SDKs CD — there are some patches at ftp://ftp.microsoft.com/developr/msdn/esqlc/)
- Microsoft Visual C++ (V4.2 or V5.0 — other C compilers might work, but we did not test them)

Follow the directions in the Administration guide when building the DLLs, except use the files in Appendix D instead of the CICS supplied source code examples.

To build the DLLs use these files in Appendix D. Be sure and read the comments thoroughly in both files before starting.

A good test when you are done and your region is up is to build, run, and install the MSSQL version of the "UXA1" transaction and CICS program to your region. This transaction can then be run from a client (or local) terminal to test the SQL Server interaction.

TIER 2 - MQLSX and MQEI

Installing

MQLSX and MQEI were installed on the Tier 2 system according the documentation in the respective readme files.

Ensure that the following environment variables are set via the Control Panel:

GMQ_XLAT_PATH = D:\MQM\MQLSX\CONV
GMQ_MQ_LIB = D:\MQM\DLL

where D: is the drive where the MQLSX source code is installed.

No problems were encountered.

Creating and starting qmgr, queues, channels

After installation, MQLSX and MQEI were configured on the Tier 2 system. In addition, for each remote Tier 3 host that would be connected to from the Tier 2, unique MQSeries objects were created and connectivity verified. From one Windows NT system, a total of four Tier 3 systems (AIX, OS/2, S/390, and Windows NT) were connected to simultaneously. For each Tier

3 host, the same series of steps were taken from command prompts; object names shown are examples of the MQLSX only:

- Started the Queue Manager with command, **STRMQM QMD1**
- Started the Listener with command, **RUNMQLSR -T TCP -M QMD1**
  Note: On AIX, the Listener is started automatically at startup instead of from a command prompt.
- Started the Channel Initiator, **RUNMQCHI -M QMD1**
- Started the Channel with command, **RUNMQCHL -M QMD1 -C D1TNTBRT**
- Started **RUNMQSC -X QMD1** to check status of channels and other objects
- From the prompt, we displayed the Sender and Receiver status by issuing the command,
  DISPLAY CHSTATUS
  If both channels (**D1TNTBRT AND NTBRTD1T**) were running successfully, then **"STATUS(RUNNING)"** would be shown for both Sender and Receiver. At that point, we were ready to test initial MQ puts and gets.

<u>Verifying the End-to-end Communications</u>
To test this connection from Windows NT Tier 2 in Austin to the Windows NT Tier 3 in Dallas, we did the following:

- On the Tier 3 host, made sure that MQSeries and MQSeries-CICS Bridge installation, configuration, and connectivity steps had been performed for each of its remote connections.
- On the AIX, OS/2, Solaris, or Windows NT side:
  - Made sure that MQSeries was started as detailed in the TIER 2 - MQLSX and MQEI Application System Section above
  - Made sure that other objects were running according to steps in TIER 2 - MQLSX and MQEI Application System above.
  - Executed sample program AMQSPUT.EXE to put a message from Tier 2 to Tier 3.
  - Executed sample program AMQSGET.EXE to get the returned message from the local queue.

<u>Hints & Tips</u>

- Use the KeepAlive parameter to avoid TCP/IP errors
  When running MQ channels over T1 lines using TCP/IP connections, we found it could be difficult to recover when TCP/IP errors occurred. The receiver's channel was getting stuck in a state of running, while the sender's channel had no status due to the error. Stopping the receiver resulted in its going into a never-ending state of stopping because it was hanging on to the TCP socket.

  We did find that stopping and restarting the Queue Manager helped us recover. Once that was done, the receiver came up in a state of running, and could then be stopped. Then the Sender and Receiver channels could be restarted and traffic would resume. A second recovery method that worked quite well was to delete the channels involved and then redefine them.

  But we found we could avoid the situation entirely by setting 'KeepAlive=yes' in the qms.ini file on all the Windows NT Queue Managers that were experiencing errors with the Sender and Receiver channels. This allowed the channels to recover without restarting Queue Managers or rebooting machines. After setting this parameter (and altering the registry on our Windows NT machines), we had no more problems with our TCP/IP channels stopping in an unresolved state. The qms.ini entry for this parameter was set as indicated below:

```
TCP:                 ; TCP/IP entries.
KeepAlive = Yes      ; Switch KeepAlive on
```

To complete setting this on a Windows NT machine, you must also alter the registry as shown below:

```
\HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\TCPIP\Parameters
KeepAlive   REG_DWORD   1
```

- MQEI configurations
  We initially used the same MQEI object configuration as we used with MQLSX. When executing the LotusScript applications, we discovered that the ReplyToQ defined on the Tier 3 host was not allowed. A feature of MQEI is that it can avoid the need for the definition of a ReplyToQ on the Tier 3 host. This method also enables the use of temporary dynamic queues for the reply.
- Set the conversion flag to "YES" for the sender channel when running MQEI
  When running MQEI applications, the conversion of data from the code character set (CCS) of the requesting system to the target system occurs at the requesting system (i.e., ASCII to EBCDIC). This was done on all Tier 2 systems using MQSeries for AIX, Solaris, OS/2, and Windows NT V5.0 accessing MQSeries V1R2 on OS/390 , MQSeries V5R0 on AIX, V5R0 on Windows NT, and V5R0 on OS/2 Warp. This was done by setting CONVERT = YES on the Sender Channels and CONVERT = NO on the Receiver Channels. Additionally, it is required to specify in the MQ Definitions database on the MQEI system the CCS of the local system and the target system. For examples, examine the sender channel definitions in Appendix B - MQSeries Object Definitions.
- IMS Retrieved Messages
  When running MQLSX application from Windows NT to S/390, messages were sent and returned; however, the content of the messages were not readable on the local message queue using the default AMQSGET.EXE provided with MQ Series. An internal program was used to browse the returned messages.
- For all MQ definitions, use UPPER case enclosed by quotes
  MQSeries is case-sensitive. All the definitions may be done using any case (or mix) that the user desires. We found that it was best to use UPPER case for all of our definitions. This way we always knew exactly how to enter the definitions, and any setup error could be spotted much more easily. We found that the best way to ensure the case on definitions is to use quotes around the names that you use. For examples of our definitions, see Appendix B - MQSeries Object Definitions.

Experiences with the Solaris Platform

MQSeries link LotusScript Extension (MQLSX)

Installing

Our installation of MQSeries on Solaris went smoothly. We used the following documentation:

- *MQSeries Messages*
- *MQSeries for Solaris Quick Beginnings*
- *MQSeries Command Reference*
- *MQSeries Application Programming Reference Summary*
- *MQSeries Application Programming Reference*
- MQLSX for Solaris documentation (MQSeries Support Pac)
- MQEI for Lotus Notes for Solaris documentation (MQSeries Support Pac)

Hints and Tips

- It is strongly recommended that the HTML or PDF softcopy versions are installed for viewing via a Web browser or printed (PDF format) if the hardcopy versions are not available. The documentation may be obtained from the CD that contains software for the product and from http://www.software.ibm.com/ts/mqseries.

Experience with application development

Application Development
To drive our scenario LotusScript applications were implemented to run from multiple Tier 2 machines on various platforms to multiple Tier 3 machines on different platforms. MQSeries Server was installed on Tier 2 and Tier 3.

MQLSX and MQEI Agents were used to access legacy data using two implementations: Notes Forms and Web browsers. The applications simulated the business logic on Tier 2. They retrieved Tier 3 database information via legacy transaction systems.

How the application works:

- The user submits input, in the form of an inventory part number, from a Notes Form or Web browser.
- The form is saved in Lotus Notes on the Tier 2 system. This triggers the local agent.
  - The criterion used to trigger agents was: Which document should the agent act on? On "documents modified or changed." To cause the triggering to be even more granular, a search was added. Only documents using the respective form, would be used to trigger the agent.
- The agent reads the input and builds an MQ message, using the appropriate format.
- The message is sent from the Tier 2 system to the Tier 3 system.
- While at the Tier 3 host, the message is retrieved, processed by the bridge application, and sent back to the local queue on the Tier 2.
- After a designated WaitTime, the agent checks the local queue for the response message.
- If present, the message is retrieved and formatted for display on Notes or Web browser.
- User checks the generated output after redisplay.

Components of MQ Message Descriptor (MQMD)

- MQLSX
  - In the LotusScript program, the agent needs knowledge of the MQSeries Queue Managers and Queue names in addition to MQMD components such as Code Character Set, Message ID, Correlation ID, Message Type, and Message Format. A full description of all components of the MQMD is given in the *MQSeries Application Programming Guide*. We specified these components as follows:

MQMsg.CharacterSet = 437
MQMsg.MessageId = String(24,0)   ' used to set the MessageId of an MQMessage to MQMI_NONE
MQCI_NEW_SESSION = "414d51214e45575f53455353494f4e5f434f5252454c4944"   'Correlation ID Setting
MQMsg.CorrelationIdHex=MQCI_NEW_SESSION
MQMsg.MessageType=MQMI_REQUEST
MQMsg.Format=MQFMT_STRING

- MQEI
  - In building the MQMD, MQEI supplies a Definition database in which the user specifies components before executing the agent. The database is read when the agent is executed to obtain needed values. From a programming perspective, this method is simpler than MQLSX because it does not require the developer to have extensive knowledge of the MQ infrastructure.
  - In Definitions Database, we used the following to specify component of the MQMD:

    ServiceType:"CICS DPL via MQSeries"

    Different Encoding and Character Sets used to connect to Hosts

MVS/ESA	500
AIX	829
Windows NT	850

Building the MQ Message:
To build the message, the MQLSX application sets the Transaction ID according the Tier 3 host's bridge application. The Transaction ID is the first part of the message read by the bridge; if it is not correct, the message is sent to the Tier 3 host's Dead Letter Queue. The second part of the message, the application data, is the input read from the Notes Form or Web browser, the inventory part number. After the message is built, the agent continues its process to put the message on the appropriate queue to send the message on its way.

Running a single agent to multiple Hosts (using spawned agents):
One of our tests included developing an agent that would call a maximum of three nested agents, also referred to as spawned agents, that would run to multiple Tier 3 systems. The intent was to present the end-user with one form in which inventory transactions could be made to three databases; otherwise, the original agent that runs to one Tier 3 host at a time would have to be executed three separate times.

To run the spawned agents to multiple Tier 3 hosts, it was required to configure all MQ objects for the three hosts. This introduced a new factor that you will not see if you're only running to one Tier 3 host — the need for multiple port numbers. We chose to create the first Queue Manager as the Default Queue Manager. By definition, its port number was 1414. For the other two Queue Managers, we had to set the port number in the appropriate MQS.INI to a unique number, i.e., 1415 and 1416, to ensure that TCP/IP traffic was not congested.

Application Deployment

- MQLSX and MQEI manual tests

  Using Lotus Notes, MQSeries, and Domino Server (for Web application) on the end-user machines,
  - message was input
  - local agent was run (The criterion used to auto-start agents was: on "documents modified or changed" - further, to make search more granular, the respective form was used based on the implementation used (Notes Form or Web browser).
  - Message sent from Tier 2 to Tier 3

- MQ checked for reply message on local queue.
- If present, message was retrieved and format for display on Notes or Web browser.
- User checked/verified the output after redisplay.

<u>Applications run to IMS on S/390 Tier 3 from Windows NT Tier 2</u>
We verified that we could successfully run MQLSX and MQEI applications to IMS on S/390.

- We ran the product supplied sample application for MQEI.
- We ran a sample MQLSX application provided by S/390 development.

<u>Error Recovery Test</u>
During our automation run, all Tier 2 systems executed MQLSX and MQEI applications to the Tier 3 systems. We selected the scenarios below to check recovery of messages:

- Using the IBM MQSeries-CICS Bridge on S/390:
  1. Stop the Tier 3 bridge on S/390, using the stop listener command. Then, through the CICS ISPF panel, force the deactivation of the bridge while messages continued to be sent from the Tier 2 servers for one minute.
  2. Stop the Queue Managers on the Tier 2 systems, using the MODE(FORCE) option, while their channels were still connected to Tier 3 on S/390.
- Using CPIT MQSeries-CICS Bridge on OS/2:
  1. Power down the Tier 2 system on Windows NT while its channels were still connected to Tier 3 on OS/2 and while messages were still being sent.
  2. Stop the Tier 2 Queue Manager on AIX, using the MODE(FORCE) option, while its channels were still connected to the OS/2 Tier 3 and while messages were still being sent.

Here is what we have found:

- In Scenario 1:
  - After the IBM MQSeries-CICS Bridge on S/390 was powered on, using the start listener command, we started the activation of the bridge through the CICS ISPF panel. The loss of connectivity to bridge resulted in massive queuing of MQ message requests (2000 messages queued on the local queue, system.cics.bridge.queue, waiting for the bridge to become active). Once the connection was restored by restarting the listener and connecting the bridge, all MQ messages were picked up. The correct CICS transactions were executed and messages sent to the appropriate Transmission Queues.
  - Because the Tier 2 Queue Managers were inactive, messages remained in the S/390 Transmission Queues.
  - The status of the S/390 Sender Channels changed from 'RUNNING' to 'RETRYING' after the Tier 2 Queue Managers were down.
  - We then restarted the Queue Manager, Listener, and Receiver Channel on the Tier 2 systems.
    - The Tier 3 Transmission Queues' status changed from "GET(DISABLED)" to "GET(ENABLED)" and the trigger settings were changed to 'TRIGGER.'
    - The 2000 messages that were in the Tier 3 Transmission Queues were successfully sent to the local queues on the Tier 2 systems.
    - All replies were received by the Tier 2 systems in the proper format.

- In Scenario 2:
  - The bridge remained active.
  - After the loss of connectivity to the Tier 2 systems:
    - 300 messages stayed in the OS/2 Transmission Queues.
    - The status of the Tier 3 Sender Channels changed from 'RUNNING' to 'RETRYING' after the Tier 2 Queue Managers were down.
    - When the Tier 3 Sender Channels had a status of 'RETRYING,' the status of their Transmission Queues changed from "GET(ENABLED)" TO "GET(DISABLED)" and the trigger settings were changed to 'NOTRIGGER.'
  - After the Tier 2 system on Windows NT was powered on, its Queue Manager, Listener, and Receiver Channel were started. For the AIX system, the Queue Manager and Receiver Channel were restarted.
  - The Tier 3 Transmission Queues' status changed from "GET(DISABLED)" to "GET(ENABLED)" and the trigger settings were changed to 'TRIGGER.'
  - The 300 messages that were in the OS/2 Transmission Queues were sent to the Tier 2 local queues on AIX and Windows NT in the proper format.

- MQSeries on S/390, OS/2, AIX and Windows NT handled the error conditions correctly .
- On the same Windows NT system, when the command was issued to start the Sender Channel, an error condition resulted:

  "Channel program started
  AMQ9558: Remote channel is not currently available
  AMQ9999: Channel program ended abnormally"

  That was due to the fact that Receiver Channel on the Tier 3 system, which corresponded to the Tier 2 Sender Channel, still maintained a status of 'Running' although the Tier 2 system was powered off. To correct the condition, we tried two methods:

  - Method 1 (Recommended):
    a. On the Tier 3, issued the following MQSC command from runmqsc session:
       STOP CHANNEL(receiver-channel-name) MODE(FORCE)
    b. On the Tier 2, started the Sender Channel by issuing the following command from an OS/2 window:
       RUNMQCHL -m (remote-qmgr-name) -c (sender-channel-name)
    c. On the Tier 3, started the Receiver Channel with MQSC command:
       START CHANNEL(receiver-channel-name)

    We learned from our experience that normally you would not stop the Receiver Channel on the Tier 3 system. Instead, only the Sender Channel on the Tier 2 should be stopped. However, if the Tier 3 Queue Manager must stop the Receiver Channel, follow steps b and c in Method 1 above to restart Sender Channel/Receiver Channel pair.

- Method 2:
  In a rare situation, Method 1 above failed to start the Sender Channel/Receiver Channel pair. To correct the condition, on Tier 3, we deleted the Receiver Channel, then redefined the same channel. The Sender/Receiver pair started successfully.

Hints and Tips

- Configuration of MQEI on the Tier 3 host requires that there must not be a ReplyToQueue, as we used with MQLSX
  - The principle behind MQEI is to support the most commonly used MQSeries functionality.
  - The standard configuration calls for the ReplyToQMgr field to always be set to blanks and the ReplyToQ to be set to EIService Inbound Connection. Further, it assumes that there is a local definition of the ReplyToQ which specifies the ReplyToQ as local, then the Domino system queue manager sets its own name in the ReplyToQMgr field.
  - However, our application called for the ReplyToQMgr to be on the Tier 3 host system, not the local. In this instance, we supplied the ReplyToQMgr name as the one on the Tier 3 host system. It then sent the message to the appropriate transmission queue and finally back to the local queue on the Tier 2 system.
- Security
  - When an EI application wishes to communicate with an enterprise application, through an EIService object, the underlying service extracts the user ID and password required from the MQEI Security database and uses these to authenticate the end-user with the enterprise system.
- MQ error 2103 (Queue Manager is currently connected....)
  When running spawned agents, a "Terminate" event is needed after each agent to clean up all current MQ objects and disconnect the Queue Manager (QM). By design, the "Terminate" is executed at the end of the unit-of-work (UOW); with the spawned agents, the UOW is the master agent which spawns the other agents. As a result, MQ object cleanup/disconnect from QM in Agent1 was not attempted until all three agents had been executed. By adding the "Terminate" event to each agent and opening the logs for Agents 2/3 for "append", instead of "write," the error was resolved.
- MQLSX Conversion problem using Web agent
  When the message was built to contain, PCSC9A95022340, the 'MQq.Put MQMsg, MQpmo' command changed the content of the message to OBRB8@84/1123/. Using a Hex table, it was clearly seen in the "before" value, PCSC9A95022340, and the "after" value OBRB8@84/1123/, that there was a difference of -1 for each of the 14 characters; i.e. P-1 = O, A-1 = @, 0-1=/, etc. We found the cause of this to be corrupted conversion tables. We replaced the following tables: 34B004E4.tbl, 04E434B0.tbl, 34B00352.tbl, and 035234B0.tbl. This corrected our problem.
- MQEI Services Definition
  - In the Services Definition of the Definition Database, the Service Context must not exceed four characters

Appendices

Appendix A - Configuration Details

Austin AIX Server (AusAIX)
Hardware Details: RS/6000 Uni-Processor
Operating System: AIX 4.3.1.0
Presentation Svcs: Motif; X-Windows; Common Desktop Environment(CDE), Netscape
    Navigator 4.06
Communications Svcs: DCE 2.1.0
App/Workgroup Svcs: Lotus Domino 4.6.2, MQSeries 5.0.0.0, MQLSX 1.3.2, MQEI 1.0
Networking: TCP/IP
Physical Network: LAN 16MB Token Ring

Austin NT Server (AusNT)
Hardware Details:
    IBM PC Server 520 8641-EZS
    CPU: 100 Mhz Pentium
    Memory - DASD: 128 MB - 2.1x4GB(RAID 1)+ 1GB
Operating System: Windows NT 4.0 + Service Pack 4 + YR2K fix
Internet Access: MS Internet Explorer V4.0
Internet Server: Lotus Domino Server V4.6.1a, Microsoft Internet Information Server V3.0
MQ Series: MQ Series V5.0, MQ LotusScript Extension(MQLSX) V1.3.2, MQ Enterprise
Integrator(MQEI) V1.0 with CSD01
Networking: TCP/IP
Physical Network: LAN 16MB Token Ring

Austin OS/2 Server (WWCS05)
Hardware Details:
    IBM PC Server 330
    CPU: 2x266 Mhz Pentium II
    Memory - DASD: 128 MB - 2x4 GB
Operating System: OS/2 WARP Server 4 SMP
Communications Svcs: IBM CICS Server for OS/2 V1.3
App/Workgroup Svcs: IBM VisualAge for Cobol V2.1, IBM VisualAge Compiler for C++
MQ Series: MQ Series V5.0, MQ LotusScript Extension(MQLSX) V1.3.2, MQ Enterprise
Integrator(MQEI) V1.0 with CSD01
Internet Access: Netscape Browser V2.02
Internet Server: Lotus Notes Domino Server 4.5
Maintenance: IBM TME Netfinity V5
Data Access Svcs: Relational Database - IBM DB2 Universal Database for OS/2V5.0
Enterprise+SDK V5
Networking: TCP/IP
Physical Network: LAN 16MB Token Ring

Austin NT Server (ausnt4)
Hardware Details:
IBM PC Server 520 8641-MD2
CPU: 4 - 200 Mhz Pentium Pro
Memory - DASD: 528 MB - (SCSI) 3x4.2 GB
Operating System: Windows NT V4.0 + Service Pack4 + YR2K fix
Communications Svcs: IBM eNetwork Communications Server V5.01
Internet Access: Microsoft Internet Explorer V4.0, Netscape Browser V4.05
Internet Server: Lotus Domino Server V4.6.1a, Microsoft Internet Integrator Server V3.0
NotesPump: Lotus NotesPump Server V2.5
MQ Series: MQ Series V5.0, MQ LotusScript Extension(MQLSX) V1.3.2, MQ Enterprise Integrator (MQEI) V1.0 with CSD01
Maintenance: IBM TME Netfinity V5.0
Networking: TCP/IP
Physical Network: LAN 16MB Token Ring
Other Apps: Microsoft FrontPage 98, Adobe Acrobat Reader V3.0

Dallas Sun Solaris Client
Hardware Details:
SPARC5 170 Mghz
2 GB Hard Drive
64 MB Memory
Operating System: Solaris 2.5.1
MQSeries: MQSeries V5 Server; MQLSX for Solaris (MA6F support pac); MQEI for Solaris
Presentation Svcs: Common Desktop Environment (CDE)
Workgroup: Domino V4.6.2; Lotus Notes Client for Solaris
Networking: TCP/IP
Physical Network: LAN 16MB Token Ring

NT Server (CPITDALNT2)
Hardware Details:
AOpen Clone PC
CPU: 1x200 Mhz Pentium
Memory - DASD: 128 MB - 1x3.2GB
Operating System: Microsoft NT Server 4.0/Service Pack 4
Communication Svcs: CICS Server 4.0.2 - service Level 2, build s400L980113
App/Workgroup Svcs: Microsoft Visual C++ V6.0
MQ Series: MQServer V5
Data Access Svcs: Relational Database - Microsoft SQL Server V6.5
Networking: TCP/IP
Physical Network: Ethernet 10Base-T

NT Server (CPITDALNT3)
Hardware Details:
AOpen Clone PC

CPU: 1x166 Mhz Pentium
Memory - DASD: 128 MB - 1x3.2GB
Operating System: Microsoft NT Server 4.0/Service Pack 3
App/Workgroup Svcs: Microsoft Visual C++ V6.0
MQ Series: MQServer V5, MQLSX V1.3.2
Workgroup Server: Notes Domino Server 4.5
Networking: TCP/IP
Physical Network: Ethernet 10Base-T

Dallas AIX (AIXCS3)
Hardware Details:
RISC System/6000 Model 390
Memory - DASD: 256 MB - 2x4.5GB
Operating System: IBM AIX 4.3.1.0
Communication Svcs: CICS Server 4.2.0.1
MQ Series: MQ Server V5.0.0.0, MQClient 5.0.0.0
Data Access Svcs: Relational Database - IBM DB2 UDB for AIX 5.2.0.0
Networking: TCP/IP
Physical Network: Token Ring High-Performance

Hardware Details: RS/6000 Uni-Processor
Operating System: AIX 4.2.1
Presentation Svcs: Motif; X-Windows; Common Desktop Environment(CDE), Browser
Networking: TCP/IP
Physical Network: (i.e., LAN 16MB Token Ring)

Poughkeepsie OS/390 and DB2 Server and IMS Server
Hardware Details: 9021-982
2.2 Terabytes of DASD
776 MB Real Storage
2 GB Expanded Storage
Operating System: OS/390 V2R7
Data Access Svcs: Relational Database- IBM DB2 for OS/390 Version 5
App/Workgroup Svcs: Domino 4.6.4, MQ V1R2, MQLSX rel 1.3.1, MQSeries-CICS/ESA Bridge Version 1.0, MQSeries-IMS/ESA Bridge Version 1.0
Networking: OS/390 eNetwork Communication Server V2R7; TCP/IP for OS/390-V2R7
Physical Network: LAN: 16Mbps Token Ring IEEE802.5, WAN: T-1 links

Poughkeepsie NT Client
Hardware Details: IBM desktop P100
CPU: 1x100 Mhz Pentium
Memory - DASD: 96 MB - 1x1.2GB
Operating System: Windows NT V4.0 + Service Pack2
Internet Access: Microsoft Internet Explorer V4.0, Netscape Browser V4.5
Networking: TCP/IP
Physical Network: LAN 16-bit Token Ring

Other Apps: Lotus Notes 4.5, Lotus SmartSuite, Adobe Acrobat Reader V3.5

<u>Rochester Host AS/400 DB2/400 (RCHASTWI)</u>
Hardware Details: AS/400 V4R2 4-Way Millennium
Operating System: OS/400 V4R3
Presentation Services: AS/400 "Green-Screen": UIM, DDS; GUI: Client Access/Unity
App/Workgroup Services: 5769LNT-Domino for AS/400 Server Version 4.5 International), and 4.6.3 (International)
Data Access Services: Relational Database - DB2/400 (DRDA, SQL, RDB), Hierarchical Database - Notes
Communications Svcs: OS/400 Networking
MQ Series: MQSeries for AS/400-part # 5769MQ2, MQSeries link LotusScript Extension (MQLSX Version 1.3.2 part# 5733MQX)
Distribution Services: AS/400 Directory and Security Services
Networking: TCP/IP, SNA/APPN
Physical Network: LAN (Token Ring)
Other Apps: Microsoft FrontPage 98, Adobe Acrobat Reader V3.0

Appendix B - MQSeries Object Definitions

This appendix provides sample MQSeries definitions from our S/390 and OS/2 systems. The definitions required for other platforms would be similar

S/390 Definitions:

```
*******************************************************************
*                                                                 *
* @START_COPYRIGHT@                                               *
*     Statement:     Licensed Materials - Property of IBM         *
*                                                                 *
*                    5695-137                                     *
*                    (C) Copyright IBM Corporation. 1993, 1996    *
*                                                                 *
*     Status:        Version 1 Release 1                          *
* @END_COPYRIGHT@                                                 *
*                                                                 *
*******************************************************************
* For MQEI make sure that the conversion flag on the sender channel*
* is set to N, and The remote queue  is commented.                *
* For MQLSX make sure that the conversion flag is set to Y on the  *
* sending channel and the remote reply queue is defined.          *
*                                                                 *
*                                                                 *
*******************************************************************
*
*
*               IBM MQSeries for MVS/ESA
* CSQINP2 sample for distributed queuing not using CICS
*
*******************************************************************
*
* In order to use the distributed queuing facilities
* that do NOT use CICS, these objects must be created the first
* time that a queue manager is started.  This can be done by
* including this data set in the CSQINP2 DD concatenation in the
* queue manager started task procedure, as shown in the sample
* procedure CSQ4MSTR.
*
* Once these objects have been successfully created there is no need
* to redefine them on subsequent queue manager starts, so this data
* set can be removed from the CSQINP2 DD concatenation.  If the data
* set is not removed from CSQINP2, the DEFINEs will fail with an
* error message saying the object already exists.  Alternatively,
* you can add the keyword REPLACE to each command if the definitions
* are to be reset for each startup.
*
*****************************************************************
*
* NOTE:  The definitions in this data set are only for
*        distributed queuing when NOT USING CICS.
*        See CSQ4DISQ for the corresponding CICS definitions.
*
***************************************************************
** CSQPMQQ Local Queue defined initially to test Raleigh's ****
**         connection to Poughkeepsie                     ****
**         client to server use CSQPMQQ reply to queue     ****
```

```
***************************************************************
DEFINE QLOCAL( 'CSQPMQQ' )  REPLACE +

* Common queue attributes
         DESCR( 'MQ traffic testing' ) +
         PUT( ENABLED ) +
         DEFPRTY( 0 ) +
         DEFPSIST( YES ) +

* Local queue attributes
         GET( ENABLED ) +
         NOSHARE +
         DEFSOPT( EXCL ) +
         MSGDLVSQ( PRIORITY ) +
         RETINTVL( 999999999 ) +
         MAXDEPTH( 999999999 ) +
         MAXMSGL( 4194304 ) +
         NOHARDENBO +
         BOTHRESH( 0 ) +
         BOQNAME( ' ' ) +
         STGCLASS( 'DEFAULT' ) +
         USAGE( NORMAL ) +

* Event control attributes
         QDPMAXEV( ENABLED ) +
         QDPHIEV( DISABLED ) +
         QDEPTHHI( 80 ) +
         QDPLOEV( DISABLED ) +
         QDEPTHLO( 40 ) +
         QSVCIEV( NONE ) +
         QSVCINT( 999999999 ) +

* Trigger attributes
         NOTRIGGER +
         TRIGTYPE( FIRST ) +
         TRIGDPTH( 1 ) +
         TRIGMPRI( 0 ) +
         TRIGDATA( ' ' ) +
         PROCESS( ' ' ) +
         INITQ( ' ' )
*
*****************************************************************
* SENDING END DEFINITIONS FOR CSQP TO ROCH AS400
*****************************************************************
*
******
DEFINE QLOCAL( 'QMC1' ) REPLACE +

* Common queue attributes
         DESCR( 'Transmission queue for QMC1' ) +
         PUT( ENABLED ) +
         DEFPRTY( 5 ) +
         DEFPSIST( YES ) +

* Local queue attributes
         GET( ENABLED ) +
         SHARE +
         DEFSOPT( EXCL ) +
         MSGDLVSQ( FIFO ) +
         RETINTVL( 999999999 ) +
         MAXDEPTH( 10000 ) +
         MAXMSGL( 4194304 ) +
         NOHARDENBO +
         BOTHRESH( 0 ) +
         BOQNAME( ' ' ) +
         STGCLASS( 'REMOTE' ) +
         USAGE( XMITQ ) +

* Event control attributes
         QDPMAXEV( ENABLED ) +
         QDPHIEV( DISABLED ) +
         QDEPTHHI( 80 ) +
         QDPLOEV( DISABLED ) +
         QDEPTHLO( 40 ) +
         QSVCIEV( NONE ) +
         QSVCINT( 999999999 ) +

* Trigger attributes
         TRIGGER +
         TRIGTYPE( FIRST ) +
         TRIGMPRI( 0 ) +
         TRIGDPTH( 1 ) +
         TRIGDATA( 'CSQPTC1T' ) +
         PROCESS( 'QMC1.SEND.PROCESS' ) +
         INITQ( 'SYSTEM.CHANNEL.INITQ' )
*
******
DEFINE PROCESS( 'QMC1.SEND.PROCESS' ) REPLACE +

* Process attributes
         DESCR( 'Process for sending messages to QMC1' ) +
         APPLTYPE( MVS ) +
         APPLICID( 'CSQX START' ) +
         USERDATA( 'CSQPTC1T' ) +
         ENVRDATA( ' ' )
*
****** CSQP (POK) TO Roch AS400 USING TCPIP
DEFINE CHANNEL( 'CSQPTC1T') +
         CHLTYPE( SDR ) +

* Sender channel attributes
         DESCR( 'Channel for sending messages to QMC1' ) +
         TRPTYPE( TCP ) REPLACE +
         XMITQ( 'QMC1' ) +
         MCAUSER( ' ' ) +
         BATCHSZ( 50 ) +
         DISCINT( 0 ) +
         SHORTRTY( 10 )       SHORTTMR( 60 ) +
         LONGRTY( 999999999 ) LONGTMR( 120 ) +
         SCYEXIT( ' ' )       SCYDATA( ' ' ) +
         MSGEXIT( ' ' )       MSGDATA( ' ' ) +
         SENDEXIT( ' ' )      SENDDATA( ' ' ) +
         RCVEXIT( ' ' )       RCVDATA( ' ' ) +
         SEQWRAP( 999999999 ) +
         CONVERT( YES ) +
         MAXMSGL( 4194304 ) +

* This defines where the target queue manager is.
* Specify either the TCP/IP network address, for example
*        CONNAME( '9.67.148.35' ) OLD IP ADDR

         CONNAME( '9.24.64.38' )

** TOC1MQQ reply to queue     **

DEFINE QREMOTE( 'TOC1MQQ' ) REPLACE +

* Common queue attributes
         DESCR( 'Queue for accessing MQQ on QMC1' ) +
         PUT( ENABLED ) +
         DEFPSIST( YES ) +
         DEFPRTY( 9 ) +

* Remote queue attributes
*  RNAME IS QUEUE ON ROCH AS400
*  RQMNAME IS QUEUE MANAGER ON AUSTIN
         RNAME( 'C1MQQ' ) +
         RQMNAME( QMC1 ) +
         XMITQ( 'QMC1' )
*
* 'TARGET.QUEUE'

*
*****************************************************************
* RECEIVING END DEFINITIONS FOR CSQP FROM ROCH AS400
*****************************************************************
*
DEFINE CHANNEL( 'C1TCSQPT') +
         CHLTYPE( RCVR ) +

* Receiver channel attributes
         DESCR( 'Channel for receiving messages from QMC1' ) +
         TRPTYPE( TCP ) REPLACE +
         BATCHSZ( 50 ) +
         SCYEXIT( ' ' )       SCYDATA( ' ' ) +
         MSGEXIT( ' ' )       MSGDATA( ' ' ) +
         SENDEXIT( ' ' )      SENDDATA( ' ' ) +
         RCVEXIT( ' ' )       RCVDATA( ' ' ) +
         MCAUSER( ' ' ) +
         PUTAUT( DEF ) +
         SEQWRAP( 999999999 ) +
         MAXMSGL( 4194304 )
*
*****************************************************************
* End of Roch AS400 Definitions
*****************************************************************
*
*********DS start Dallas SOLARIS*********************************
*****************************************************************
*****************************************************************
* SENDING END DEFINITIONS FOR CSQP TO DALLAS SOLARIS
*****************************************************************
*
******
DEFINE QLOCAL( 'QMDS' ) REPLACE +

* Common queue attributes
```

```
         DESCR( 'Transmission queue for QMDS' ) +
         PUT( ENABLED ) +
         DEFPRTY( 5 ) +
         DEFPSIST( YES ) +

* Local queue attributes
         GET( ENABLED ) +
         SHARE +
         DEFSOPT( EXCL ) +
         MSGDLVSQ( FIFO ) +
         RETINTVL( 999999999 ) +
         MAXDEPTH( 10000 ) +
         MAXMSGL( 4194304 ) +
         NOHARDENBO +
         BOTHRESH( 0 ) +
         BOQNAME( ' ' ) +
         STGCLASS( 'REMOTE' ) +
         USAGE( XMITQ ) +

* Event control attributes
         QDPMAXEV( ENABLED ) +
         QDPHIEV( DISABLED ) +
         QDEPTHHI( 80 ) +
         QDPLOEV( DISABLED ) +
         QDEPTHLO( 40 ) +
         QSVCIEV( NONE ) +
         QSVCINT( 999999999 ) +

* Trigger attributes
         TRIGGER +
         TRIGTYPE( FIRST ) +
         TRIGMPRI( 0 ) +
         TRIGDPTH( 1 ) +
         TRIGDATA( 'CSQPTDST' ) +
         PROCESS( 'QMDS.SEND.PROCESS' ) +
         INITQ( 'SYSTEM.CHANNEL.INITQ' )
*
******
DEFINE PROCESS( 'QMDS.SEND.PROCESS' ) REPLACE +

* Process attributes
         DESCR( 'Process for sending messages to QMDS' ) +
         APPLTYPE( MVS ) +
         APPLICID( 'CSQX START' ) +
         USERDATA( 'CSQPTDST' ) +
         ENVRDATA( ' ' )
*
****** CSQP (ME) TO DALLAS SOLARIS USING TCPIP
DEFINE CHANNEL( 'CSQPTDST') +
         CHLTYPE( SDR ) +

* Sender channel attributes
         DESCR( 'Channel for sending messages to QMDS' ) +
         TRPTYPE( TCP ) REPLACE +
         XMITQ( 'QMDS' ) +
         MCAUSER( ' ' ) +
         BATCHSZ( 50 ) +
         DISCINT( 0 ) +
         SHORTRTY( 10 )       SHORTTMR( 60 ) +
         LONGRTY( 999999999 ) LONGTMR( 120 ) +
         SCYEXIT( ' ' )       SCYDATA( ' ' ) +
         MSGEXIT( ' ' )       MSGDATA( ' ' ) +
         SENDEXIT( ' ' )      SENDDATA( ' ' ) +
         RCVEXIT( ' ' )       RCVDATA( ' ' ) +
         SEQWRAP( 999999999 ) +
         CONVERT( YES ) +
         MAXMSGL( 4194304 ) +

* This defines where the target queue manager is.
* Specify either the TCP/IP network address, for example
*        CONNAME( '9.67.148.35' ) OLD IP ADDR
         CONNAME( '9.24.34.59(1414)' )
**use TODSMQQ reply to queue     **
DEFINE QREMOTE( 'TODSMQQ' ) REPLACE +

* Common queue attributes
         DESCR( 'Queue for accessing MQQ on QMDS' ) +
         PUT( ENABLED ) +
         DEFPSIST( YES ) +
         DEFPRTY( 9 ) +

* Remote queue attributes
*  RNAME IS QUEUE ON DALLAS
*  RQMNAME IS QUEUE MANAGER ON DALLAS
         RNAME( 'DSMQQ' ) +
         RQMNAME( QMDS ) +
         XMITQ( 'QMDS' )
*
* 'TARGET.QUEUE'
* also needs to be defined as a LOCAL queue on QMDS.
*
*
*****************************************************************
* RECEIVING END DEFINITIONS FOR CSQP FROM DALLAS SOLARIS
*****************************************************************
*
DEFINE CHANNEL( 'DSTCSQPT') +
         CHLTYPE( RCVR ) +

* Receiver channel attributes
         DESCR( 'Channel for receiving messages from QMDS' ) +
         TRPTYPE( TCP ) REPLACE +
         BATCHSZ( 50 ) +
         SCYEXIT( ' ' )       SCYDATA( ' ' ) +
         MSGEXIT( ' ' )       MSGDATA( ' ' ) +
         SENDEXIT( ' ' )      SENDDATA( ' ' ) +
         RCVEXIT( ' ' )       RCVDATA( ' ' ) +
         MCAUSER( ' ' ) +
         PUTAUT( DEF ) +
         SEQWRAP( 999999999 ) +
         MAXMSGL( 4194304 )
*
*****************************************************************
* End of DALLAS  Definitions
*****************************************************************
*
*********DI start Dallas SOLARIS  for MQEI**********************
*****************************************************************
```

```
*
* SENDING END DEFINITIONS FOR CSQP TO DALLAS SOLARIS
*****************************************************************
*
******
DEFINE QLOCAL( 'QMDI' ) REPLACE +

* Common queue attributes
         DESCR( 'Transmission queue for QMDI' ) +
         PUT( ENABLED ) +
         DEFPRTY( 5 ) +
         DEFPSIST( YES ) +

* Local queue attributes
         GET( ENABLED ) +
         SHARE +
         DEFSOPT( EXCL ) +
         MSGDLVSQ( FIFO ) +
         RETINTVL( 999999999 ) +
         MAXDEPTH( 10000 ) +
         MAXMSGL( 4194304 ) +
         NOHARDENBO +
         BOTHRESH( 0 ) +
         BOQNAME( ' ' ) +
         STGCLASS( 'REMOTE' ) +
         USAGE( XMITQ ) +

* Event control attributes
         QDPMAXEV( ENABLED ) +
         QDPHIEV( DISABLED ) +
         QDEPTHHI( 80 ) +
         QDPLOEV( DISABLED ) +
         QDEPTHLO( 40 ) +
         QSVCIEV( NONE ) +
         QSVCINT( 999999999 ) +

* Trigger attributes
         TRIGGER +
         TRIGTYPE( FIRST ) +
         TRIGMPRI( 0 ) +
         TRIGDPTH( 1 ) +
         TRIGDATA( 'CSQPTDIT' ) +
         PROCESS( 'QMDI.SEND.PROCESS' ) +
         INITQ( 'SYSTEM.CHANNEL.INITQ' )
*
******
DEFINE PROCESS( 'QMDI.SEND.PROCESS' ) REPLACE +

* Process attributes
         DESCR( 'Process for sending messages to QMDI' ) +
         APPLTYPE( MVS ) +
         APPLICID( 'CSQX START' ) +
         USERDATA( 'CSQPTDIT' ) +
         ENVRDATA( ' ' )
*
****** CSQP (ME) TO DALLAS SOLARIS USING TCPIP
DEFINE CHANNEL( 'CSQPTDIT') +
         CHLTYPE( SDR ) +
* Sender channel attributes
         DESCR( 'Channel for sending messages to QMDI' ) +
         TRPTYPE( TCP ) REPLACE +
         XMITQ( 'QMDI' ) +
         MCAUSER( ' ' ) +
         BATCHSZ( 50 ) +
         DISCINT( 0 ) +
         SHORTRTY( 10 )       SHORTTMR( 60 ) +
         LONGRTY( 999999999 ) LONGTMR( 120 ) +
         SCYEXIT( ' ' )       SCYDATA( ' ' ) +
         MSGEXIT( ' ' )       MSGDATA( ' ' ) +
         SENDEXIT( ' ' )      SENDDATA( ' ' ) +
         RCVEXIT( ' ' )       RCVDATA( ' ' ) +
         SEQWRAP( 999999999 ) +
         CONVERT( NO ) +
         MAXMSGL( 4194304 ) +

* This defines where the target queue manager is.
* Specify either the TCP/IP network address, for example
*        CONNAME( '9.67.148.35' ) OLD IP ADDR
         CONNAME( '9.24.34.59(1415)' )
*
** don't need reply to queue for MQEI MGR   ****
* DEFINE QREMOTE( 'TODIMQQ' ) REPLACE +

* Common queue attributes
*        DESCR( 'Queue for accessing MQQ on QMDI' ) +
*        PUT( ENABLED ) +
*        DEFPSIST( YES ) +
*        DEFPRTY( 9 ) +
*
* Remote queue attributes
*  RNAME IS QUEUE ON DALLAS
*  RQMNAME IS QUEUE MANAGER ON DALLAS
*        RNAME( 'DIMQQ' ) +
*        RQMNAME( QMDI ) +
*        XMITQ( 'QMDI' )
*
* 'TARGET.QUEUE'
* also needs to be defined as a LOCAL queue on QMDI.
*
*
*****************************************************************
* RECEIVING END DEFINITIONS FOR CSQP FROM DALLAS SOLARIS
*****************************************************************
*
DEFINE CHANNEL( 'DITCSQPT') +
         CHLTYPE( RCVR ) +

* Receiver channel attributes
         DESCR( 'Channel for receiving messages from QMDI' ) +
         TRPTYPE( TCP ) REPLACE +
         BATCHSZ( 50 ) +
         SCYEXIT( ' ' )       SCYDATA( ' ' ) +
         MSGEXIT( ' ' )       MSGDATA( ' ' ) +
         SENDEXIT( ' ' )      SENDDATA( ' ' ) +
         RCVEXIT( ' ' )       RCVDATA( ' ' ) +
         MCAUSER( ' ' ) +
         PUTAUT( DEF ) +
         SEQWRAP( 999999999 ) +
         MAXMSGL( 4194304 )
*
*****************************************************************
* End of DALLAS  Definitions for MQEI
*****************************************************************
*
**********DN start Dallas NT*************************************
*****************************************************************
*****************************************************************
* SENDING END DEFINITIONS FOR CSQP TO DALLAS NT
*****************************************************************
*
******
DEFINE QLOCAL( 'QMDN' ) REPLACE +

* Common queue attributes
         DESCR( 'Transmission queue for QMDN' ) +
         PUT( ENABLED ) +
         DEFPRTY( 5 ) +
         DEFPSIST( YES ) +

* Local queue attributes
         GET( ENABLED ) +
         SHARE +
         DEFSOPT( EXCL ) +
         MSGDLVSQ( FIFO ) +
         RETINTVL( 999999999 ) +
         MAXDEPTH( 10000 ) +
         MAXMSGL( 4194304 ) +
         NOHARDENBO +
         BOTHRESH( 0 ) +
         BOQNAME( ' ' ) +
         STGCLASS( 'REMOTE' ) +
         USAGE( XMITQ ) +

* Event control attributes
         QDPMAXEV( ENABLED ) +
         QDPHIEV( DISABLED ) +
         QDEPTHHI( 80 ) +
         QDPLOEV( DISABLED ) +
         QDEPTHLO( 40 ) +
         QSVCIEV( NONE ) +
         QSVCINT( 999999999 ) +

* Trigger attributes
         TRIGGER +
         TRIGTYPE( FIRST ) +
         TRIGMPRI( 0 ) +
         TRIGDPTH( 1 ) +
         TRIGDATA( 'CSQPTDNT' ) +
         PROCESS( 'QMDN.SEND.PROCESS' ) +
         INITQ( 'SYSTEM.CHANNEL.INITQ' )
*
******
DEFINE PROCESS( 'QMDN.SEND.PROCESS' ) REPLACE +

* Process attributes
         DESCR( 'Process for sending messages to QMDN' ) +
         APPLTYPE( MVS ) +
         APPLICID( 'CSQX START' ) +
         USERDATA( 'CSQPTDNT' ) +
         ENVRDATA( ' ' )
*
****** CSQP (ME) TO DALLAS NT USING TCPIP
DEFINE CHANNEL( 'CSQPTDNT') +
         CHLTYPE( SDR ) +

* Sender channel attributes
         DESCR( 'Channel for sending messages to QMDN' ) +
         TRPTYPE( TCP ) REPLACE +
         XMITQ( 'QMDN' ) +
         MCAUSER( ' ' ) +
         BATCHSZ( 50 ) +
         DISCINT( 0 ) +
         SHORTRTY( 10 )       SHORTTMR( 60 ) +
         LONGRTY( 999999999 ) LONGTMR( 120 ) +
         SCYEXIT( ' ' )       SCYDATA( ' ' ) +
         MSGEXIT( ' ' )       MSGDATA( ' ' ) +
         SENDEXIT( ' ' )      SENDDATA( ' ' ) +
         RCVEXIT( ' ' )       RCVDATA( ' ' ) +
         SEQWRAP( 999999999 ) +
         CONVERT( YES ) +
         MAXMSGL( 4194304 ) +

* This defines where the target queue manager is.
* Specify either the TCP/IP network address, for example
*        CONNAME( '9.67.148.35' ) OLD IP ADDR
         CONNAME( '9.24.49.9' )
** use TODNMQQ reply to queue     ****
DEFINE QREMOTE( 'TODNMQQ' ) REPLACE +

* Common queue attributes
         DESCR( 'Queue for accessing MQQ on QMDN' ) +
         PUT( ENABLED ) +
         DEFPSIST( YES ) +
         DEFPRTY( 9 ) +

* Remote queue attributes
*  RNAME IS QUEUE ON DALLAS
*  RQMNAME IS QUEUE MANAGER ON DALLAS
         RNAME( 'DNMQQ' ) +
         RQMNAME( QMDN ) +
         XMITQ( 'QMDN' )
*
* 'TARGET.QUEUE'
* also needs to be defined as a LOCAL queue on QMDN.
*
*
*****************************************************************
* RECEIVING END DEFINITIONS FOR CSQP FROM DALLAS NT
*****************************************************************
*
DEFINE CHANNEL( 'DNTCSQPT') +
         CHLTYPE( RCVR ) +

* Receiver channel attributes
         DESCR( 'Channel for receiving messages from QMDN' ) +
         TRPTYPE( TCP ) REPLACE +
         BATCHSZ( 50 ) +
         SCYEXIT( ' ' )       SCYDATA( ' ' ) +
         MSGEXIT( ' ' )       MSGDATA( ' ' ) +
         SENDEXIT( ' ' )      SENDDATA( ' ' ) +
         RCVEXIT( ' ' )       RCVDATA( ' ' ) +
         MCAUSER( ' ' ) +
         PUTAUT( DEF ) +
         SEQWRAP( 999999999 ) +
         MAXMSGL( 4194304 )
*
*****************************************************************
* End of Austin Definitions
*****************************************************************
*
*****************************************************************
******DN end DALLAS NT ******************************************
*****************************************************************
*
*****************************************************************
**********A1 start Austin NT*************************************
*****************************************************************
*
*****************************************************************
* SENDING END DEFINITIONS FOR CSQP TO AUSTIN NT
*****************************************************************
*
******
DEFINE QLOCAL( 'QMA1' ) REPLACE +

* Common queue attributes
         DESCR( 'Transmission queue for QMA1' ) +
         PUT( ENABLED ) +
         DEFPRTY( 5 ) +
         DEFPSIST( YES ) +

* Local queue attributes
         GET( ENABLED ) +
         SHARE +
         DEFSOPT( EXCL ) +
         MSGDLVSQ( FIFO ) +
         RETINTVL( 999999999 ) +
         MAXDEPTH( 10000 ) +
         MAXMSGL( 4194304 ) +
         NOHARDENBO +
         BOTHRESH( 0 ) +
         BOQNAME( ' ' ) +
         STGCLASS( 'REMOTE' ) +
         USAGE( XMITQ ) +

* Event control attributes
         QDPMAXEV( ENABLED ) +
         QDPHIEV( DISABLED ) +
         QDEPTHHI( 80 ) +
         QDPLOEV( DISABLED ) +
         QDEPTHLO( 40 ) +
         QSVCIEV( NONE ) +
         QSVCINT( 999999999 ) +
* Trigger attributes
         TRIGGER +
         TRIGTYPE( FIRST ) +
         TRIGMPRI( 0 ) +
         TRIGDPTH( 1 ) +
         TRIGDATA( 'CSQPTA1T' ) +
         PROCESS( 'QMA1.SEND.PROCESS' ) +
         INITQ( 'SYSTEM.CHANNEL.INITQ' )
*
******
DEFINE PROCESS( 'QMA1.SEND.PROCESS' ) REPLACE +

* Process attributes
         DESCR( 'Process for sending messages to QMA1' ) +
         APPLTYPE( MVS ) +
         APPLICID( 'CSQX START' ) +
         USERDATA( 'CSQPTA1T' ) +
         ENVRDATA( ' ' )
*
****** CSQP (ME) TO AUSTIN NT USING TCPIP
DEFINE CHANNEL( 'CSQPTA1T') +
         CHLTYPE( SDR ) +

* Sender channel attributes
         DESCR( 'Channel for sending messages to QMA1' ) +
         TRPTYPE( TCP ) REPLACE +
         XMITQ( 'QMA1' ) +
         MCAUSER( ' ' ) +
         BATCHSZ( 50 ) +
         DISCINT( 0 ) +
         SHORTRTY( 10 )       SHORTTMR( 60 ) +
         LONGRTY( 999999999 ) LONGTMR( 120 ) +
         SCYEXIT( ' ' )       SCYDATA( ' ' ) +
         MSGEXIT( ' ' )       MSGDATA( ' ' ) +
         SENDEXIT( ' ' )      SENDDATA( ' ' ) +
         RCVEXIT( ' ' )       RCVDATA( ' ' ) +
         SEQWRAP( 999999999 ) +
         CONVERT( YES ) +
         MAXMSGL( 4194304 ) +

* This defines where the target queue manager is.
* Specify either the TCP/IP network address, for example
*        CONNAME( '9.67.148.35' ) OLD IP ADDR
         CONNAME( '9.24.65.61(1414)' )
**  use TOA1MQQ reply to queue     ****
DEFINE QREMOTE( 'TOA1MQQ' ) REPLACE +

* Common queue attributes
         DESCR( 'Queue for accessing MQQ on QMA1' ) +
         PUT( ENABLED ) +
         DEFPSIST( YES ) +
         DEFPRTY( 9 ) +

* Remote queue attributes
*  RNAME IS QUEUE ON AUSTIN
*  RQMNAME IS QUEUE MANAGER ON AUSTIN
         RNAME( 'A1MQQ' ) +
         RQMNAME( QMA1 ) +
         XMITQ( 'QMA1' )
```

```
*
* 'TARGET.QUEUE'
* also needs to be defined as a LOCAL queue on QMA1.
*
*
*****************************************************************
* RECEIVING END DEFINITIONS FOR CSQP FROM AUSTIN NT
*****************************************************************
*
DEFINE CHANNEL( 'A1TCSQPT') +
         CHLTYPE( RCVR ) +

* Receiver channel attributes
         DESCR( 'Channel for receiving messages from QMA1' ) +
         TRPTYPE( TCP ) REPLACE +
         BATCHSZ( 50 ) +
         SCYEXIT( ' ' )       SCYDATA( ' ' ) +
         MSGEXIT( ' ' )       MSGDATA( ' ' ) +
         SENDEXIT( ' ' )      SENDDATA( ' ' ) +
         RCVEXIT( ' ' )       RCVDATA( ' ' ) +
         MCAUSER( ' ' ) +
         PUTAUT( DEF ) +
         SEQWRAP( 999999999 ) +
         MAXMSGL( 4194304 )
*
*****************************************************************
* End of Austin Definitions
*****************************************************************
*
*****************************************************************
******A1 end Austin NT ******************************************
*****************************************************************
*
*****************************************************************
* SENDING END DEFINITIONS FOR CSQP TO AUSTIN AIX- MQEI
*****************************************************************
*
******
DEFINE QLOCAL( 'QMM3' ) REPLACE +

* Common queue attributes
         DESCR( 'Transmission queue for QMM3' ) +
         PUT( ENABLED ) +
         DEFPRTY( 5 ) +
         DEFPSIST( YES ) +

* Local queue attributes
         GET( ENABLED ) +
         SHARE +
         DEFSOPT( EXCL ) +
         MSGDLVSQ( FIFO ) +
         RETINTVL( 999999999 ) +
         MAXDEPTH( 10000 ) +
         MAXMSGL( 4194304 ) +
         NOHARDENBO +
         BOTHRESH( 0 ) +
         BOQNAME( ' ' ) +
         STGCLASS( 'REMOTE' ) +
         USAGE( XMITQ ) +
* Event control attributes
         QDPMAXEV( ENABLED ) +
         QDPHIEV( DISABLED ) +
         QDEPTHHI( 80 ) +
         QDPLOEV( DISABLED ) +
         QDEPTHLO( 40 ) +
         QSVCIEV( NONE ) +
         QSVCINT( 999999999 ) +

* Trigger attributes
         TRIGGER +
         TRIGTYPE( FIRST ) +
         TRIGMPRI( 0 ) +
         TRIGDPTH( 1 ) +
         TRIGDATA( 'CSQPTM3T' ) +
         PROCESS( 'QMM3.SEND.PROCESS' ) +
         INITQ( 'SYSTEM.CHANNEL.INITQ' )
*
******
DEFINE PROCESS( 'QMM3.SEND.PROCESS' ) REPLACE +

* Process attributes
         DESCR( 'Process for sending messages to QMM3' ) +
         APPLTYPE( MVS ) +
         APPLICID( 'CSQX START' ) +
         USERDATA( 'CSQPTM3T' ) +
         ENVRDATA( ' ' )
*
****** CSQP (ME) TO AUSTIN AIX MQEI USING TCPIP
DEFINE CHANNEL( 'CSQPTM3T') +
         CHLTYPE( SDR ) +

* Sender channel attributes
         DESCR( 'Channel for sending messages to QMM3' ) +
         TRPTYPE( TCP ) REPLACE +
         XMITQ( 'QMM3' ) +
         MCAUSER( ' ' ) +
         BATCHSZ( 50 ) +
         DISCINT( 0 ) +
         SHORTRTY( 10 )       SHORTTMR( 60 ) +
         LONGRTY( 999999999 ) LONGTMR( 120 ) +
         SCYEXIT( ' ' )       SCYDATA( ' ' ) +
         MSGEXIT( ' ' )       MSGDATA( ' ' ) +
         SENDEXIT( ' ' )      SENDDATA( ' ' ) +
         RCVEXIT( ' ' )       RCVDATA( ' ' ) +
         SEQWRAP( 999999999 ) +
         CONVERT( NO ) +
         MAXMSGL( 4194304 ) +

* This defines where the target queue manager is.
* Specify either the TCP/IP network address, for example
*        CONNAME( '9.67.148.35' ) OLD IP ADDR
         CONNAME( '9.24.65.2(1415)' )
*
*****************************************************************
* RECEIVING END DEFINITIONS FOR CSQP FROM AUSTIN AIX MQEI
*****************************************************************
*
```

```
DEFINE CHANNEL( 'M3TCSQPT') +
         CHLTYPE( RCVR ) +

* Receiver channel attributes
         DESCR( 'Channel for receiving messages from QMM3' ) +
         TRPTYPE( TCP ) REPLACE +
         BATCHSZ( 50 ) +
         SCYEXIT( ' ' )       SCYDATA( ' ' ) +
         MSGEXIT( ' ' )       MSGDATA( ' ' ) +
         SENDEXIT( ' ' )      SENDDATA( ' ' ) +
         RCVEXIT( ' ' )       RCVDATA( ' ' ) +
         MCAUSER( ' ' ) +
         PUTAUT( DEF ) +
         SEQWRAP( 999999999 ) +
         MAXMSGL( 4194304 )
*
*****************************************************************
* End of Austin AIX Definitions for MQEI
*****************************************************************
*
*****************************************************************
* SENDING END DEFINITIONS FOR CSQP TO AUSTIN NT - MQEI
*****************************************************************
*
******
DEFINE QLOCAL( 'QMM1' ) REPLACE +

* Common queue attributes
         DESCR( 'Transmission queue for QMM1' ) +
         PUT( ENABLED ) +
         DEFPRTY( 5 ) +
         DEFPSIST( YES ) +

* Local queue attributes
         GET( ENABLED ) +
         SHARE +
         DEFSOPT( EXCL ) +
         MSGDLVSQ( FIFO ) +
         RETINTVL( 999999999 ) +
         MAXDEPTH( 10000 ) +
         MAXMSGL( 4194304 ) +
         NOHARDENBO +
         BOTHRESH( 0 ) +
         BOQNAME( ' ' ) +
         STGCLASS( 'REMOTE' ) +
         USAGE( XMITQ ) +

* Event control attributes
         QDPMAXEV( ENABLED ) +
         QDPHIEV( DISABLED ) +
         QDEPTHHI( 80 ) +
         QDPLOEV( DISABLED ) +
         QDEPTHLO( 40 ) +
         QSVCIEV( NONE ) +
         QSVCINT( 999999999 ) +

* Trigger attributes
         TRIGGER +
         TRIGTYPE( FIRST ) +
         TRIGMPRI( 0 ) +
         TRIGDPTH( 1 ) +
         TRIGDATA( 'CSQPTM1T' ) +
         PROCESS( 'QMM1.SEND.PROCESS' ) +
         INITQ( 'SYSTEM.CHANNEL.INITQ' )
*
******
DEFINE PROCESS( 'QMM1.SEND.PROCESS' ) REPLACE +

* Process attributes
         DESCR( 'Process for sending messages to QMM1' ) +
         APPLTYPE( MVS ) +
         APPLICID( 'CSQX START' ) +
         USERDATA( 'CSQPTM1T' ) +
         ENVRDATA( ' ' )
*
****** CSQP (ME) TO AUSTIN #1 USING TCPIP
DEFINE CHANNEL( 'CSQPTM1T') +
         CHLTYPE( SDR ) +

* Sender channel attributes
         DESCR( 'Channel for sending messages to QMM1' ) +
         TRPTYPE( TCP ) REPLACE +
         XMITQ( 'QMM1' ) +
         MCAUSER( ' ' ) +
         BATCHSZ( 50 ) +
         DISCINT( 0 ) +
         SHORTRTY( 10 )       SHORTTMR( 60 ) +
         LONGRTY( 999999999 ) LONGTMR( 120 ) +
         SCYEXIT( ' ' )       SCYDATA( ' ' ) +
         MSGEXIT( ' ' )       MSGDATA( ' ' ) +
         SENDEXIT( ' ' )      SENDDATA( ' ' ) +
         RCVEXIT( ' ' )       RCVDATA( ' ' ) +
         SEQWRAP( 999999999 ) +
         CONVERT( NO ) +
         MAXMSGL( 4194304 ) +

* This defines where the target queue manager is.
* Specify either the TCP/IP network address, for example
*        CONNAME( '9.67.148.35' ) OLD IP ADDR
         CONNAME( '9.24.65.61(1421)' )
*
*****************************************************************
* RECEIVING END DEFINITIONS FOR CSQP FROM AUSTIN NT MQEI
*****************************************************************
*
DEFINE CHANNEL( 'M1TCSQPT') +
         CHLTYPE( RCVR ) +

* Receiver channel attributes
         DESCR( 'Channel for receiving messages from QMM1' ) +
         TRPTYPE( TCP ) REPLACE +
         BATCHSZ( 50 ) +
         SCYEXIT( ' ' )       SCYDATA( ' ' ) +
         MSGEXIT( ' ' )       MSGDATA( ' ' ) +
         SENDEXIT( ' ' )      SENDDATA( ' ' ) +
         RCVEXIT( ' ' )       RCVDATA( ' ' ) +
         MCAUSER( ' ' ) +
         PUTAUT( DEF ) +
         SEQWRAP( 999999999 ) +
         MAXMSGL( 4194304 )
*
*****************************************************************
* End of Austin Definitions for MQEI
*****************************************************************
*
*****************************************************************
* SENDING END DEFINITIONS FOR CSQP TO AUSTIN OS2- MQEI
*****************************************************************
*
******
DEFINE QLOCAL( 'QMOS' ) REPLACE +

* Common queue attributes
         DESCR( 'Transmission queue for QMOS' ) +
         PUT( ENABLED ) +
         DEFPRTY( 5 ) +
         DEFPSIST( YES ) +

* Local queue attributes
         GET( ENABLED ) +
         SHARE +
         DEFSOPT( EXCL ) +
         MSGDLVSQ( FIFO ) +
         RETINTVL( 999999999 ) +
         MAXDEPTH( 10000 ) +
         MAXMSGL( 4194304 ) +
         NOHARDENBO +
         BOTHRESH( 0 ) +
         BOQNAME( ' ' ) +
         STGCLASS( 'REMOTE' ) +
         USAGE( XMITQ ) +

* Event control attributes
         QDPMAXEV( ENABLED ) +
         QDPHIEV( DISABLED ) +
         QDEPTHHI( 80 ) +
         QDPLOEV( DISABLED ) +
         QDEPTHLO( 40 ) +
         QSVCIEV( NONE ) +
         QSVCINT( 999999999 ) +

* Trigger attributes
         TRIGGER +
         TRIGTYPE( FIRST ) +
         TRIGMPRI( 0 ) +
         TRIGDPTH( 1 ) +
         TRIGDATA( 'CSQPTOST' ) +
         PROCESS( 'QMOS.SEND.PROCESS' ) +
         INITQ( 'SYSTEM.CHANNEL.INITQ' )
*
******
DEFINE PROCESS( 'QMOS.SEND.PROCESS' ) REPLACE +

* Process attributes
         DESCR( 'Process for sending messages to QMOS' ) +
         APPLTYPE( MVS ) +
         APPLICID( 'CSQX START' ) +
         USERDATA( 'CSQPTOST' ) +
         ENVRDATA( ' ' )
*
****** CSQP (ME) TO AUSTIN OS2 USING TCPIP
DEFINE CHANNEL( 'CSQPTOST') +
         CHLTYPE( SDR ) +

* Sender channel attributes
         DESCR( 'Channel for sending messages to QMOS' ) +
         TRPTYPE( TCP ) REPLACE +
         XMITQ( 'QMOS' ) +
         MCAUSER( ' ' ) +
         BATCHSZ( 50 ) +
         DISCINT( 0 ) +
         SHORTRTY( 10 )       SHORTTMR( 60 ) +
         LONGRTY( 999999999 ) LONGTMR( 120 ) +
         SCYEXIT( ' ' )       SCYDATA( ' ' ) +
         MSGEXIT( ' ' )       MSGDATA( ' ' ) +
         SENDEXIT( ' ' )      SENDDATA( ' ' ) +
         RCVEXIT( ' ' )       RCVDATA( ' ' ) +
         SEQWRAP( 999999999 ) +
         CONVERT( NO ) +
         MAXMSGL( 4194304 ) +

* This defines where the target queue manager is.
* Specify either the TCP/IP network address, for example
*        CONNAME( '9.67.148.35' ) OLD IP ADDR
         CONNAME( '9.24.65.58(1414)' )
*
*****************************************************************
* RECEIVING END DEFINITIONS FOR CSQP FROM AUSTIN NT MQEI
*****************************************************************
*
DEFINE CHANNEL( 'OSTCSQPT') +
         CHLTYPE( RCVR ) +

* Receiver channel attributes
         DESCR( 'Channel for receiving messages from QMOS' ) +
         TRPTYPE( TCP ) REPLACE +
         BATCHSZ( 50 ) +
         SCYEXIT( ' ' )       SCYDATA( ' ' ) +
         MSGEXIT( ' ' )       MSGDATA( ' ' ) +
         SENDEXIT( ' ' )      SENDDATA( ' ' ) +
         RCVEXIT( ' ' )       RCVDATA( ' ' ) +
         MCAUSER( ' ' ) +
         PUTAUT( DEF ) +
         SEQWRAP( 999999999 ) +
         MAXMSGL( 4194304 )
*
*****************************************************************
* End of Austin Definitions for MQEI
*****************************************************************
*****************************************************************
* SENDING END DEFINITIONS FOR CSQP TO AUSTIN OS2 --- MQLSX
*****************************************************************
*
******
DEFINE QLOCAL( 'QMO1' ) REPLACE +
* Common queue attributes
         DESCR( 'Transmission queue for QMO1' ) +
         PUT( ENABLED ) +
         DEFPRTY( 5 ) +
         DEFPSIST( YES ) +

* Local queue attributes
         GET( ENABLED ) +
         SHARE +
         DEFSOPT( EXCL ) +
         MSGDLVSQ( FIFO ) +
         RETINTVL( 999999999 ) +
         MAXDEPTH( 10000 ) +
         MAXMSGL( 4194304 ) +
         NOHARDENBO +
         BOTHRESH( 0 ) +
         BOQNAME( ' ' ) +
         STGCLASS( 'REMOTE' ) +
         USAGE( XMITQ ) +

* Event control attributes
         QDPMAXEV( ENABLED ) +
         QDPHIEV( DISABLED ) +
         QDEPTHHI( 80 ) +
         QDPLOEV( DISABLED ) +
         QDEPTHLO( 40 ) +
         QSVCIEV( NONE ) +
         QSVCINT( 999999999 ) +

* Trigger attributes
         TRIGGER +
         TRIGTYPE( FIRST ) +
         TRIGMPRI( 0 ) +
         TRIGDPTH( 1 ) +
         TRIGDATA( 'CSQPTO1T' ) +
         PROCESS( 'QMO1.SEND.PROCESS' ) +
         INITQ( 'SYSTEM.CHANNEL.INITQ' )
*
******
DEFINE PROCESS( 'QMO1.SEND.PROCESS' ) REPLACE +

* Process attributes
         DESCR( 'Process for sending messages to QMO1' ) +
         APPLTYPE( MVS ) +
         APPLICID( 'CSQX START' ) +
         USERDATA( 'CSQPTO1T' ) +
         ENVRDATA( ' ' )
*
****** CSQP (ME) TO AUSTIN #1 USING TCPIP
DEFINE CHANNEL( 'CSQPTO1T') +
         CHLTYPE( SDR ) +

* Sender channel attributes
         DESCR( 'Channel for sending messages to QMO1' ) +
         TRPTYPE( TCP ) REPLACE +
         XMITQ( 'QMO1' ) +
         MCAUSER( ' ' ) +
         BATCHSZ( 50 ) +
         DISCINT( 0 ) +
         SHORTRTY( 10 )       SHORTTMR( 60 ) +
         LONGRTY( 999999999 ) LONGTMR( 120 ) +
         SCYEXIT( ' ' )       SCYDATA( ' ' ) +
         MSGEXIT( ' ' )       MSGDATA( ' ' ) +
         SENDEXIT( ' ' )      SENDDATA( ' ' ) +
         RCVEXIT( ' ' )       RCVDATA( ' ' ) +
         SEQWRAP( 999999999 ) +
         CONVERT( YES ) +
         MAXMSGL( 4194304 ) +

* This defines where the target queue manager is.
* Specify either the TCP/IP network address, for example
*        CONNAME( '9.67.148.35' ) OLD IP ADDR
         CONNAME( '9.24.65.58(1422)' )
DEFINE QREMOTE( 'TOO1MQQ' ) REPLACE +

* Common queue attributes
         DESCR( 'Queue for accessing MQQ on QMO1' ) +
         PUT( ENABLED ) +
         DEFPSIST( YES ) +
         DEFPRTY( 9 ) +

* Remote queue attributes
*  RNAME IS QUEUE ON AUSTIN
*  RQMNAME IS QUEUE MANAGER ON AUSTIN
         RNAME( 'O1MQQ' ) +
         RQMNAME( QMO1 ) +
         XMITQ( 'QMO1' )
*







*
*****************************************************************
* RECEIVING END DEFINITIONS FOR CSQP FROM AUSTIN OS2
*****************************************************************
*
DEFINE CHANNEL( 'O1TCSQPT') +
         CHLTYPE( RCVR ) +

* Receiver channel attributes
         DESCR( 'Channel for receiving messages from QMO1' ) +
         TRPTYPE( TCP ) REPLACE +
         BATCHSZ( 50 ) +
         SCYEXIT( ' ' )       SCYDATA( ' ' ) +
         MSGEXIT( ' ' )       MSGDATA( ' ' ) +
         SENDEXIT( ' ' )      SENDDATA( ' ' ) +
         RCVEXIT( ' ' )       RCVDATA( ' ' ) +
         MCAUSER( ' ' ) +
         PUTAUT( DEF ) +
         SEQWRAP( 999999999 ) +
         MAXMSGL( 4194304 )
*
*****************************************************************
* End of Austin Definitions
```

```
*****************************************************************
*****************************************************************
* SENDING END DEFINITIONS FOR CSQP TO AUSTIN AIX --- MQLSX
*****************************************************************
*
******
DEFINE QLOCAL( 'QMX1' ) REPLACE +

* Common queue attributes
         DESCR( 'Transmission queue for QMX1' ) +
         PUT( ENABLED ) +
         DEFPRTY( 5 ) +
         DEFPSIST( YES ) +

* Local queue attributes
         GET( ENABLED ) +
         SHARE +
         DEFSOPT( EXCL ) +
         MSGDLVSQ( FIFO ) +
         RETINTVL( 999999999 ) +
         MAXDEPTH( 10000 ) +
         MAXMSGL( 4194304 ) +
         NOHARDENBO +
         BOTHRESH( 0 ) +
         BOQNAME( ' ' ) +
         STGCLASS( 'REMOTE' ) +
         USAGE( XMITQ ) +

* Event control attributes
         QDPMAXEV( ENABLED ) +
         QDPHIEV( DISABLED ) +
         QDEPTHHI( 80 ) +
         QDPLOEV( DISABLED ) +
         QDEPTHLO( 40 ) +
         QSVCIEV( NONE ) +
         QSVCINT( 999999999 ) +

* Trigger attributes
         TRIGGER +
         TRIGTYPE( FIRST ) +
         TRIGMPRI( 0 ) +
         TRIGDPTH( 1 ) +
         TRIGDATA( 'CSQPTX1T' ) +
         PROCESS( 'QMX1.SEND.PROCESS' ) +
         INITQ( 'SYSTEM.CHANNEL.INITQ' )
*
******
DEFINE PROCESS( 'QMX1.SEND.PROCESS' ) REPLACE +

* Process attributes
         DESCR( 'Process for sending messages to QMX1' ) +
         APPLTYPE( MVS ) +
         APPLICID( 'CSQX START' ) +
         USERDATA( 'CSQPTX1T' ) +
         ENVRDATA( ' ' )
*
****** CSQP (ME) TO AUSTIN #1 USING TCPIP
DEFINE CHANNEL( 'CSQPTX1T') +
         CHLTYPE( SDR ) +
* Sender channel attributes
         DESCR( 'Channel for sending messages to QMX1' ) +
         TRPTYPE( TCP ) REPLACE +
         XMITQ( 'QMX1' ) +
         MCAUSER( ' ' ) +
         BATCHSZ( 50 ) +
         DISCINT( 0 ) +
         SHORTRTY( 10 )       SHORTTMR( 60 ) +
         LONGRTY( 999999999 ) LONGTMR( 120 ) +
         SCYEXIT( ' ' )       SCYDATA( ' ' ) +
         MSGEXIT( ' ' )       MSGDATA( ' ' ) +
         SENDEXIT( ' ' )      SENDDATA( ' ' ) +
         RCVEXIT( ' ' )       RCVDATA( ' ' ) +
         SEQWRAP( 999999999 ) +
         CONVERT( YES ) +
         MAXMSGL( 4194304 ) +

* This defines where the target queue manager is.
* Specify either the TCP/IP network address, for example
*        CONNAME( '9.67.148.35' ) OLD IP ADDR
         CONNAME( '9.24.65.2(1414)' )
** use TOX1MQQ reply to queue     ****
DEFINE QREMOTE( 'TOX1MQQ' ) REPLACE +

* Common queue attributes
         DESCR( 'Queue for accessing MQQ on QMX1' ) +
         PUT( ENABLED ) +
         DEFPSIST( YES ) +
         DEFPRTY( 9 ) +

* Remote queue attributes
*  RNAME IS QUEUE ON AUSTIN
*  RQMNAME IS QUEUE MANAGER ON AUSTIN
         RNAME( 'X1MQQ' ) +
         RQMNAME( QMX1 ) +
         XMITQ( 'QMX1' )
*
* 'TARGET.QUEUE'
* also needs to be defined as a LOCAL queue on QMX1.
*
*
*****************************************************************
* RECEIVING END DEFINITIONS FOR CSQP FROM AUSTIN AIX
*****************************************************************
*
DEFINE CHANNEL( 'X1TCSQPT') +
         CHLTYPE( RCVR ) +

* Receiver channel attributes
         DESCR( 'Channel for receiving messages from QMX1' ) +
         TRPTYPE( TCP ) REPLACE +
         BATCHSZ( 50 ) +
         SCYEXIT( ' ' )       SCYDATA( ' ' ) +
         MSGEXIT( ' ' )       MSGDATA( ' ' ) +
         SENDEXIT( ' ' )      SENDDATA( ' ' ) +
         RCVEXIT( ' ' )       RCVDATA( ' ' ) +
         MCAUSER( ' ' ) +
         PUTAUT( DEF ) +
         SEQWRAP( 999999999 ) +
         MAXMSGL( 4194304 )
*
*****************************************************************
* End of Austin Definitions  AIX
***** IMS BRIDGE DEFINITIONS                      *****
************************************************************
 DEFINE QLOCAL('SYSTEM.IMS.BRIDGE.QUEUE') REPLACE +
        DESCR('IMS BRIDGE REQUEST QUEUE') +
*                                          Permit shared access
         SHARE +
*                                          FIFO Delivery
         MSGDLVSQ(FIFO) +
*                                          Persistent
         DEFPSIST(YES) +
         STGCLASS( 'IMSPETZ0' ) +
*                                          Backout hardened
         HARDENBO
************************************************************
***** CICS BRIDGE DEFINITIONS                      *****
************************************************************
 DEFINE QLOCAL('SYSTEM.CICS.BRIDGE.QUEUE') REPLACE +
        DESCR('CICS BRIDGE REQUEST QUEUE') +
*                                          Permit shared access
         SHARE +
*                                          FIFO Delivery
         MSGDLVSQ(FIFO) +
*                                          Persistent
         DEFPSIST(YES) +
*                                          Backout hardened
         HARDENBO +
*                                          Trigger on 1st message
*                                          using default initq
         TRIGGER TRIGTYPE(FIRST) +
         PROCESS('CICS_BRIDGE') +
         INITQ('CICS01.INITQ')
********************************************************************
 DEFINE PROCESS('CICS_BRIDGE') REPLACE +
        DESCR('CICS BRIDGE MONITOR') +
        APPLICID('CKBR') +
        APPLTYPE(CICS) +
```

OS/2 Definitions:

<u>Here are the MQSeries Definitions for MQLSX on QMN2 Queue Manager for NT defined on OS/2 :</u>

```
*******************************************************************/
*             OS/2 BRIDGE DEFINITIONS FOR CPIT LAB                */
*             Notes Domino to OS/2 Host server                    */
*               Creation date :  10-28-1998                       */
*               Change History:  11-20-1998                       */
*                                                                 */
*******************************************************************/
*   (C) Copyright International Business Machines Corp. 1997,1998  */
*;  All Rights Reserved                                           */
*;  US Government Users Restricted Rights - Use, duplication or    */
*;  disclosure restricted by GSA ADP Schedule Contract with IBM    */
*;  Corp.                                                         */
*;                                                                */
*   Licensed Materials - Property of IBM                          */
*;                                                                */
*;                                                                */
*;     NOTICE TO USERS OF THE SOURCE CODE EXAMPLES                */
*;                                                                */
*; INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THE SOURCE */
*; CODE EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, "AS */
*; IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED,  */
*; INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF        */
*; MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.  THE     */
*; ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE SOURCE CODE*/
*; EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, IS WITH  */
*; YOU.  SHOULD ANY PART OF THE SOURCE CODE EXAMPLES PROVE        */
*; DEFECTIVE, YOU (AND NOT IBM OR AN AUTHORIZED DEALER) ASSUME THE */
*; ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.   */
*;                                                                */
*;                                                                */
*                                                                 */
*******************************************************************/
*******************************************************************/
*              HUB FOR OS/2 MQLSX: NT QMGR AND OS/2 BRIDGE        */
*                                                                 */
* Remote Queue Manager at AusNT: QMN2                             */
* Local Queue Manager at OS/2 Bridge: OSBR                        */
*******************************************************************/

*******************************************************************/
*                    DEFINE LOCAL QUEUE                           */
*******************************************************************/


    DEFINE QLOCAL('OS.DLQ') REPLACE       +
       DESCR('Dead Letter Queue')      +
       DEFPSIST(YES)             +
       MAXDEPTH(640000)                +
       MAXMSGL(4194304)                +
       SHARE                     +
       USAGE(NORMAL)             +
       TRIGTYPE(NONE)
    DEFINE QLOCAL('LEGACY.LOCAL') REPLACE +
       DESCR('INPUT QUEUE')           +
       DEFPSIST(YES)            +
       MAXDEPTH(640000)               +
       MAXMSGL(4194304)               +
       SHARE                    +
       USAGE(NORMAL)            +
       TRIGTYPE(NONE)


    DEFINE QLOCAL('CLIENT.LOCAL') REPLACE +
       DESCR('OUTPUT Queue')          +
       DEFPSIST(YES)            +
       MAXDEPTH(640000)               +
       MAXMSGL(4194304)               +
       SHARE                    +
       USAGE(NORMAL)            +
       TRIGTYPE(NONE)


  DEFINE QLOCAL('SYSTEM.CICS.BRIDGE.QUEUE') REPLACE   +
       DESCR('MQ general message queue')        +
       DEFPSIST(YES)                     +
       MAXDEPTH(640000)                        +
       MAXMSGL(4194304)                        +
       SHARE                             +
       USAGE(NORMAL)                     +
       TRIGTYPE(NONE)

********************************************************************/
*                        MESSAGE QUEUE                             */
********************************************************************/

* QMN2 REMOTE QUEUE *

   DEFINE QREMOTE('TON2MQQ') REPLACE   +
         DESCR('messages to QMN2')     +
         PUT(ENABLED)                  +
         DEFPRTY(0)              +
         DEFPSIST(YES)                 +
         XMITQ('QMN2')                 +
         RNAME('N2MQQ')                +
         RQMNAME('QMN2')         +
         SCOPE(QMGR)

********************************************************************/
*                CHANNEL DEFINITIONS over TCP                      */
* start runmqchl -c OSBRTN2T -m OSBR                               */
* start runmqlsr -t tcp -m OSBR                                    */
********************************************************************/

* OSBR & QMN2 CHANNEL pair*

   DEFINE CHANNEL('OSBRTN2T') CHLTYPE(SDR)            +
         TRPTYPE(TCP) REPLACE DESCR(' ')        +
         BATCHSZ(4) DISCINT(6000)               +
         CONNAME('9.24.65.61(1414)') XMITQ('QMN2')    +
         SHORTTMR(60) SHORTRTY(10)              +
         LONGTMR(1200) LONGRTY(999999999)       +
         SEQWRAP(999999999) MAXMSGL(4194304)          +
         CONVERT(YES)

   DEFINE CHANNEL('N2TOSBRT') CHLTYPE(RCVR)           +
         TRPTYPE(TCP) REPLACE DESCR(' ')        +
         BATCHSZ(50)                      +
         PUTAUT(DEF) SEQWRAP(999999999)         +
         MAXMSGL(4194304)

******************************************************************/
*                     DEFINE LOCAL "XMITQ" QUEUE                 */
******************************************************************/

   DEFINE QLOCAL('QMN2') REPLACE          +
         DESCR('messages to QMN2')        +
         DEFPSIST(YES)                    +
         MAXDEPTH(640000)                 +
         MAXMSGL(4194304)                 +
         SHARE                         +
         USAGE(XMITQ)                  +
         INITQ(SYSTEM.CHANNEL.INITQ) TRIGGER    +
         TRIGTYPE(FIRST)                  +
         TRIGDATA('OSBRTN2T')

******************************************************************/
* >>>>>>>>>>>>                                 <<<<<<<<<<<<<<<<< */
* >>>>>>>    END OF SETUP: MQLSX - NT TO OS/2 BRIDGE       <<<<<<< */
* >>>>>>>>>>>>                                 <<<<<<<<<<<<<<<<< */
******************************************************************/
```

Here are the MQSeries Definitions for MQEI on QMN3 Queue Manager for NT defined on OS/2:

```
******************************************************************/
*             OS/2 BRIDGE DEFINITIONS FOR CPIT LAB               */
*             Notes Domino to OS/2 Host server                   */
*               Creation date :  10-28-1998                      */
*               Change History:  11-20-1998                      */
*                                                                */
******************************************************************/
*   (C) Copyright International Business Machines Corp. 1997,1998  */
*;  All Rights Reserved                                          */
*;  US Government Users Restricted Rights - Use, duplication or    */
*;  disclosure restricted by GSA ADP Schedule Contract with IBM    */
*;  Corp.                                                        */
*;                                                               */
*   Licensed Materials - Property of IBM                         */
*;                                                               */
*;                                                               */
*;     NOTICE TO USERS OF THE SOURCE CODE EXAMPLES            */
*;                                                               */
*; INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THE SOURCE */
*; CODE EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, "AS */
*; IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED,  */
*; INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF        */
*; MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.  THE      */
*; ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE SOURCE CODE*/
*; EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, IS WITH  */
*; YOU.  SHOULD ANY PART OF THE SOURCE CODE EXAMPLES PROVE         */
*; DEFECTIVE, YOU (AND NOT IBM OR AN AUTHORIZED DEALER) ASSUME THE */
*; ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.   */
*;                                                                 */
*;                                                                 */
*                                                                  */
*******************************************************************/
*******************************************************************/
*              HUB FOR OS/2 MQEI: NT QMGR AND OS/2 BRIDGE          */
*                                                                  */
* Remote Queue Manager at AusNT: QMN3                              */
* Local Queue Manager at OS/2 Bridge: OSBR                         */
*******************************************************************/
*******************************************************************/

*******************************************************************/
*                    DEFINE LOCAL QUEUE                            */
*******************************************************************/

    DEFINE QLOCAL('OS.DLQ') REPLACE       +
       DESCR('Dead Letter Queue')      +
       DEFPSIST(YES)             +
       MAXDEPTH(640000)                +
       MAXMSGL(4194304)                +
       SHARE                     +
       USAGE(NORMAL)             +
       TRIGTYPE(NONE)

    DEFINE QLOCAL('LEGACY.LOCAL') REPLACE +
       DESCR('INPUT QUEUE')            +
       DEFPSIST(YES)             +
       MAXDEPTH(640000)                +
       MAXMSGL(4194304)                +
       SHARE                     +
       USAGE(NORMAL)             +
       TRIGTYPE(NONE)


    DEFINE QLOCAL('CLIENT.LOCAL') REPLACE +
       DESCR('OUTPUT Queue')           +
       DEFPSIST(YES)             +
       MAXDEPTH(640000)                +
       MAXMSGL(4194304)                +
       SHARE                     +
       USAGE(NORMAL)             +
       TRIGTYPE(NONE)

  DEFINE QLOCAL('SYSTEM.CICS.BRIDGE.QUEUE') REPLACE   +
       DESCR('MQ general message queue')        +
       DEFPSIST(YES)                      +
       MAXDEPTH(640000)                         +
       MAXMSGL(4194304)                         +
       SHARE                              +
       USAGE(NORMAL)                      +
       TRIGTYPE(NONE)

*******************************************************************/
*               CHANNEL DEFINITIONS over TCP                       */
* start runmqchl -c OSBRTN3T -m OSBR                        */
* start runmqlsr -t tcp -m OSBR                                  */
*******************************************************************/

* OSBR and QMN3 pair *

   DEFINE CHANNEL('OSBRTN3T') CHLTYPE(SDR)            +
         TRPTYPE(TCP) REPLACE DESCR(' ')        +
         BATCHSZ(4) DISCINT(6000)               +
         CONNAME('9.24.65.61(1415)') XMITQ('QMN3')    +
         SHORTTMR(60) SHORTRTY(10)              +
         LONGTMR(1200) LONGRTY(999999999)       +
         SEQWRAP(999999999) MAXMSGL(4194304)          +
         CONVERT(NO)

   DEFINE CHANNEL('N3TOSBRT') CHLTYPE(RCVR)     +
         TRPTYPE(TCP) REPLACE DESCR(' ')  +
         BATCHSZ(50)                   +
         PUTAUT(DEF) SEQWRAP(999999999)   +
         MAXMSGL(4194304)

*******************************************************************/
*                   DEFINE LOCAL "XMITQ" QUEUE                     */
*******************************************************************/

   DEFINE QLOCAL('QMN3') REPLACE          +
       DESCR('messages to QMN3')          +
       DEFPSIST(YES)                   +
       MAXDEPTH(640000)                   +
       MAXMSGL(4194304)                   +
       SHARE                           +
      USAGE(XMITQ)                     +
       INITQ(SYSTEM.CHANNEL.INITQ) TRIGGER      +
       TRIGTYPE(FIRST)                    +
       TRIGDATA('OSBRTN3T')

*******************************************************************/
* >>>>>>>>>>>>                                   <<<<<<<<<<<<<<<<< */
* >>>>>>>    END OF SETUP: MQEI - NT TO OS/2 BRIDGE        <<<<<<< */
* >>>>>>>>>>>>                                   <<<<<<<<<<<<<<<<< */
*******************************************************************/
```

<u>Here are the MQSeries Definitions for MQLSX on OSBR Queue Manager for OS/2 defined on Austin NT :</u>

```
*******************************************************************/
*             OS/2 BRIDGE DEFINITIONS FOR CPIT LAB                 */
*             Notes Domino to OS/2 Host server                     */
*               Creation date :  10-28-1998                        */
*               Change History:  11-20-1998                        */
*                                                                  */
*******************************************************************/
*   (C) Copyright International Business Machines Corp. 1997,1998  */
*;  All Rights Reserved                                            */
*;  US Government Users Restricted Rights - Use, duplication or    */
*;  disclosure restricted by GSA ADP Schedule Contract with IBM    */
*;  Corp.                                                            */
*;                                                                   */
*   Licensed Materials - Property of IBM                             */
*;                                                                   */
*;                                                                   */
*;     NOTICE TO USERS OF THE SOURCE CODE EXAMPLES                   */
*;                                                                   */
*; INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THE SOURCE */
*; CODE EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, "AS */
*; IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED,  */
*; INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF         */
*; MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.  THE      */
*; ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE SOURCE CODE*/
*; EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, IS WITH  */
*; YOU.  SHOULD ANY PART OF THE SOURCE CODE EXAMPLES PROVE         */
*; DEFECTIVE, YOU (AND NOT IBM OR AN AUTHORIZED DEALER) ASSUME THE */
*; ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.   */
*;                                                                   */
*;                                                                   */
*                                                                    */
*******************************************************************/
*******************************************************************/
*              HUB FOR OS/2 MQLSX: OS/2 QMGR DEFINITIONS ON NT     */
*                                                                  */
* Remote Queue Manager at OS/2: OSBR                               */
* Local Queue Manager at NT: QMN2                                  */
*******************************************************************/

*******************************************************************/
*                    DEFINE LOCAL QUEUE                            */
*******************************************************************/

    DEFINE QLOCAL('N2.DLQ') REPLACE       +
       DESCR('Dead Letter Queue')         +
       DEFPSIST(YES)                      +
       MAXDEPTH(640000)                   +
       MAXMSGL(4194304)                   +
       SHARE                              +
       USAGE(NORMAL)                      +
       TRIGTYPE(NONE)

    DEFINE QLOCAL('N2MQQ') REPLACE        +
    DESCR('MQ general message queue')     +
       DEFPSIST(YES)                      +
       MAXDEPTH(640000)                   +
       MAXMSGL(4194304)                   +
       SHARE                              +
       USAGE(NORMAL)                      +
       TRIGTYPE(NONE)

*******************************************************************/
*                       MESSAGE QUEUE                              */
*******************************************************************/

* OSBR REMOTE QUEUE *

   DEFINE QREMOTE('TO.SYSTEM.CICS.BRIDGE.QUEUE') REPLACE    +
         DESCR('messages to OSBR')                          +
         PUT(ENABLED)                                       +
         DEFPRTY(0)                                    +
         DEFPSIST(YES)                                 +
         XMITQ('OSBR')                                 +
         RNAME('SYSTEM.CICS.BRIDGE.QUEUE')             +
         RQMNAME('OSBR')                               +
         SCOPE(QMGR)

*******************************************************************/
*              CHANNEL DEFINITIONS over TCP                       */
* start runmqchl -c N2TOSBRT -m QMN2                              */
* start runmqlsr -t tcp -m QMN2                                   */
*******************************************************************/

* OSBR & QMN2 CHANNEL pair *

   DEFINE CHANNEL('N2TOSBRT') CHLTYPE(SDR)             +
         TRPTYPE(TCP) REPLACE DESCR(' ')               +
         BATCHSZ(4) DISCINT(6000)                      +
         CONNAME('9.24.65.58(1420)') XMITQ('OSBR')     +
         SHORTTMR(60) SHORTRTY(10)                     +
         LONGTMR(1200) LONGRTY(999999999)              +
         SEQWRAP(999999999) MAXMSGL(4194304)           +
         CONVERT(NO)

   DEFINE CHANNEL('OSBRTN2T') CHLTYPE(RCVR)      +
         TRPTYPE(TCP) REPLACE DESCR(' ')         +
         BATCHSZ(50)                             +
         PUTAUT(DEF) SEQWRAP(999999999)          +
         MAXMSGL(4194304)

*******************************************************************/
*                    DEFINE LOCAL "XMITQ" QUEUE                   */
*******************************************************************/

   DEFINE QLOCAL('OSBR') REPLACE           +
         DESCR('messages to OSBR')         +
         DEFPSIST(YES)                     +
         MAXDEPTH(640000)                  +
         MAXMSGL(4194304)                  +
         SHARE                             +
         USAGE(XMITQ)                      +
         INITQ(SYSTEM.CHANNEL.INITQ) TRIGGER    +
         TRIGTYPE(FIRST)                   +
         TRIGDATA('N2TOSBRT')

*******************************************************************/
* >>>>>>>>>>>>                                  <<<<<<<<<<<<<<<<< */
* >>>>>>>    END OF SETUP: MQLSX - NT TO OS/2 BRIDGE       <<<<<<< */
* >>>>>>>>>>>>                                  <<<<<<<<<<<<<<<<< */
*******************************************************************/
```

<u>Here are the MQSeries Definitions for MQEI on OSBR Queue Manager for OS/2 defined on Austin NT:</u>

```
*******************************************************************/
*            OS/2 BRIDGE DEFINITIONS FOR CPIT LAB                 */
*             Notes Domino to OS/2 Host server                    */
*               Creation date :  10-28-1998                       */
*               Change History:  11-20-1998                       */
*                                                                 */
*******************************************************************/
*   (C) Copyright International Business Machines Corp. 1997,1998  */
*;  All Rights Reserved                                            */
*;  US Government Users Restricted Rights - Use, duplication or    */
*;  disclosure restricted by GSA ADP Schedule Contract with IBM    */
*;  Corp.                                                          */
*;                                                                 */
*   Licensed Materials - Property of IBM                           */
*;                                                                 */
*;                                                                 */
*;     NOTICE TO USERS OF THE SOURCE CODE EXAMPLES                 */
*;                                                                 */
*; INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THE SOURCE */
*; CODE EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, "AS */
*; IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED,  */
*; INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF         */
*; MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.  THE      */
*; ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE SOURCE CODE*/
*; EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, IS WITH  */
*; YOU.  SHOULD ANY PART OF THE SOURCE CODE EXAMPLES PROVE         */
*; DEFECTIVE, YOU (AND NOT IBM OR AN AUTHORIZED DEALER) ASSUME THE */
*; ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.   */
*;                                                                 */
*;                                                                 */
*                                                                  */
*******************************************************************/
*******************************************************************/
*              HUB FOR OS/2 MQEI: OS/2 QMGR DEFINITIONS ON NT      */
*                                                                  */
* Remote Queue Manager at OS/2: OSBR                               */
* Local Queue Manager at NT: QMN3                                  */
*******************************************************************/

*******************************************************************/
*                    DEFINE LOCAL QUEUE                            */
*******************************************************************/

    DEFINE QLOCAL('N3.DLQ') REPLACE       +
       DESCR('Dead Letter Queue')         +
       DEFPSIST(YES)                      +
       MAXDEPTH(640000)                   +
       MAXMSGL(4194304)                   +
       SHARE                              +
       USAGE(NORMAL)                      +
       TRIGTYPE(NONE)

    DEFINE QLOCAL('N3MQQ') REPLACE        +
    DESCR('MQ general message queue')     +
       DEFPSIST(YES)                      +
       MAXDEPTH(640000)                   +
       MAXMSGL(4194304)                   +
       SHARE                              +
       USAGE(NORMAL)                      +
       TRIGTYPE(NONE)
```

```
*******************************************************************/
*                        MESSAGE QUEUE                            */
*******************************************************************/

* OSBR REMOTE QUEUE *

   DEFINE QREMOTE('TO.SYSTEM.CICS.BRIDGE.QUEUE') REPLACE    +
         DESCR('messages to OSBR')                          +
         PUT(ENABLED)                                       +
         DEFPRTY(0)                                         +
         DEFPSIST(YES)                                      +
         XMITQ('OSBR')                                      +
         RNAME('SYSTEM.CICS.BRIDGE.QUEUE')                  +
         RQMNAME('OSBR')                                    +
         SCOPE(QMGR)

*******************************************************************/
*                CHANNEL DEFINITIONS over TCP                     */
* start runmqchl -c N3TOSBRT -m QMN3                              */
* start runmqlsr -t tcp -m QMN3                                   */
*******************************************************************/

* OSBR & QMN3 CHANNEL pair *

   DEFINE CHANNEL('N3TOSBRT') CHLTYPE(SDR)                  +
         TRPTYPE(TCP) REPLACE DESCR(' ')                    +
         BATCHSZ(4) DISCINT(6000)                           +
         CONNAME('9.24.65.58(1420)') XMITQ('OSBR')          +
         SHORTTMR(60) SHORTRTY(10)                          +
         LONGTMR(1200) LONGRTY(999999999)                   +
         SEQWRAP(999999999) MAXMSGL(4194304)                +
         CONVERT(YES)

   DEFINE CHANNEL('OSBRTN3T') CHLTYPE(RCVR)           +
         TRPTYPE(TCP) REPLACE DESCR(' ')              +
         BATCHSZ(50)                                  +
         PUTAUT(DEF) SEQWRAP(999999999)               +
         MAXMSGL(4194304)

*******************************************************************/
*                      DEFINE LOCAL "XMITQ" QUEUE                 */
*******************************************************************/

   DEFINE QLOCAL('OSBR') REPLACE                +
         DESCR('messages to OSBR')              +
         DEFPSIST(YES)                          +
         MAXDEPTH(640000)                       +
         MAXMSGL(4194304)                       +
         SHARE                                  +
         USAGE(XMITQ)                           +
         INITQ(SYSTEM.CHANNEL.INITQ) TRIGGER    +
         TRIGTYPE(FIRST)                        +
         TRIGDATA('N3TOSBRT')

*******************************************************************/
* >>>>>>>>>>>>                                  <<<<<<<<<<<<<<<<< */
* >>>>>>>    END OF SETUP: MQEI - NT TO OS/2 BRIDGE       <<<<<<<  */
* >>>>>>>>>>>>                                  <<<<<<<<<<<<<<<<< */
*******************************************************************/
```

Here are the MQSeries Definitions for MQLSX on QMX2 Queue Manager for AIX defined on OS/2:

```
*******************************************************************/
*             OS/2 BRIDGE DEFINITIONS FOR CPIT LAB                */
*             Notes Domino to OS/2 Host server                    */
*               Creation date :  10-28-1998                       */
*               Change History:  11-20-1998                       */
*                                                                 */
*******************************************************************/
*   (C) Copyright International Business Machines Corp. 1997,1998  */
*;  All Rights Reserved                                           */
*;  US Government Users Restricted Rights - Use, duplication or   */
*;  disclosure restricted by GSA ADP Schedule Contract with IBM   */
*;  Corp.                                                         */
*;                                                                */
*   Licensed Materials - Property of IBM                          */
*;                                                                */
*;                                                                */
*;     NOTICE TO USERS OF THE SOURCE CODE EXAMPLES            */
*;                                                                */
*; INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THE SOURCE */
*; CODE EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, "AS */
*; IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED,  */
*; INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF        */
*; MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.  THE     */
*; ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE SOURCE CODE*/
*; EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, IS WITH  */
*; YOU.  SHOULD ANY PART OF THE SOURCE CODE EXAMPLES PROVE        */
*; DEFECTIVE, YOU (AND NOT IBM OR AN AUTHORIZED DEALER) ASSUME THE */
*; ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.   */
*;                                                                */
*;                                                                */
*                                                                 */
*******************************************************************/
*******************************************************************/
*              HUB FOR OS/2 MQLSX: AIX QMGR AND OS/2 BRIDGE       */
*                                                                 */
* Remote Queue Manager at AIX: QMX2                               */
* Local Queue Manager at OS/2 Bridge: OSBR                        */
*******************************************************************/

*******************************************************************/

*******************************************************************/
*                    DEFINE LOCAL QUEUE                           */
*******************************************************************/

    DEFINE QLOCAL('OS.DLQ') REPLACE       +
       DESCR('Dead Letter Queue')      +
       DEFPSIST(YES)             +
       MAXDEPTH(640000)                +
       MAXMSGL(4194304)                +
       SHARE                     +
       USAGE(NORMAL)             +
       TRIGTYPE(NONE)

    DEFINE QLOCAL('LEGACY.LOCAL') REPLACE +
       DESCR('INPUT QUEUE')           +
       DEFPSIST(YES)            +
       MAXDEPTH(640000)               +
       MAXMSGL(4194304)               +
       SHARE                    +
       USAGE(NORMAL)            +
       TRIGTYPE(NONE)


    DEFINE QLOCAL('CLIENT.LOCAL') REPLACE +
       DESCR('OUTPUT Queue')          +
       DEFPSIST(YES)            +
       MAXDEPTH(640000)               +
       MAXMSGL(4194304)               +
       SHARE                    +
       USAGE(NORMAL)            +
       TRIGTYPE(NONE)

  DEFINE QLOCAL('SYSTEM.CICS.BRIDGE.QUEUE') REPLACE   +
       DESCR('MQ general message queue')        +
       DEFPSIST(YES)                      +
       MAXDEPTH(640000)                         +
       MAXMSGL(4194304)                         +
       SHARE                              +
       USAGE(NORMAL)                      +
       TRIGTYPE(NONE)

********************************************************************/
*                      RESPONSE QUEUE                               */
********************************************************************/

* QMX2 REMOTE QUEUE *

   DEFINE QREMOTE('TOX2MQQ') REPLACE   +
         DESCR('messages to OSBR')     +
         PUT(ENABLED)                  +
         DEFPRTY(0)              +
         DEFPSIST(YES)                 +
         XMITQ('QMX2')                 +
         RNAME('X2MQQ')                +
         RQMNAME('QMX2')         +
         SCOPE(QMGR)

********************************************************************/
*               CHANNEL DEFINITIONS over TCP                        */
* start runmqchl -c OSBRTX2T -m OSBR                                */
* start runmqlsr -t tcp -m OSBR                                     */
********************************************************************/

* OSBR and QMX2 pair *

   DEFINE CHANNEL('OSBRTX2T') CHLTYPE(SDR)             +
         TRPTYPE(TCP) REPLACE DESCR(' ')         +
         BATCHSZ(4) DISCINT(6000)                +
         CONNAME('9.24.65.2(1487)') XMITQ('QMX2')      +
         SHORTTMR(60) SHORTRTY(10)               +
         LONGTMR(1200) LONGRTY(999999999)       +
         SEQWRAP(999999999) MAXMSGL(4194304)          +
         CONVERT(YES)

   DEFINE CHANNEL('X2TOSBRT') CHLTYPE(RCVR)     +
         TRPTYPE(TCP) REPLACE DESCR(' ')  +
         BATCHSZ(50)                   +
         PUTAUT(DEF) SEQWRAP(999999999)   +
         MAXMSGL(4194304)

********************************************************************/
*                 DEFINE LOCAL "XMITQ" QUEUE                        */
********************************************************************/

   DEFINE QLOCAL('QMX2') REPLACE          +
         DESCR('messages to QMX2')        +
         DEFPSIST(YES)                    +
         MAXDEPTH(640000)              +
         MAXMSGL(4194304)              +
         SHARE                         +
         USAGE(XMITQ)                     +
         INITQ(SYSTEM.CHANNEL.INITQ) TRIGGER    +
         TRIGTYPE(FIRST)               +
         TRIGDATA('OSBRTX2T')

********************************************************************/
* >>>>>>>>>>>>                                  <<<<<<<<<<<<<<<<< */
* >>>>>>>      END OF SETUP: MQLSX - AIX TO OSBR           <<<<<<< */
* >>>>>>>>>>>>                                  <<<<<<<<<<<<<<<<< */
********************************************************************/
```

<u>Here are the MQSeries Definitions for MQEI on QMX3 Queue Manager for AIX defined on OS/2:</u>

```
********************************************************************/
*                  MQEI: AIX QMGR AND OS/2 BRIDGE                  */
*                                                                  */
*     Queue Managers: Remote (AIX): QMX3; Local (OS/2): OSBR       */
********************************************************************/

********************************************************************/
*                    DEFINE LOCAL QUEUE                            */
********************************************************************/

    DEFINE QLOCAL('OS.DLQ') REPLACE       +
       DESCR('Dead Letter Queue')      +
       DEFPSIST(YES)             +
       MAXDEPTH(640000)                +
       MAXMSGL(4194304)                +
       SHARE                     +
       USAGE(NORMAL)             +
       TRIGTYPE(NONE)

    DEFINE QLOCAL('LEGACY.LOCAL') REPLACE +
       DESCR('INPUT QUEUE')            +
       DEFPSIST(YES)             +
       MAXDEPTH(640000)                +
       MAXMSGL(4194304)                +
       SHARE                     +
       USAGE(NORMAL)             +
       TRIGTYPE(NONE)


    DEFINE QLOCAL('CLIENT.LOCAL') REPLACE +
       DESCR('OUTPUT Queue')           +
       DEFPSIST(YES)             +
       MAXDEPTH(640000)                +
       MAXMSGL(4194304)                +
       SHARE                     +
       USAGE(NORMAL)             +
       TRIGTYPE(NONE)

  DEFINE QLOCAL('SYSTEM.CICS.BRIDGE.QUEUE') REPLACE   +
       DESCR('MQ general message queue')        +
       DEFPSIST(YES)                      +
       MAXDEPTH(640000)                         +
       MAXMSGL(4194304)                         +
       SHARE                              +
       USAGE(NORMAL)                      +
       TRIGTYPE(NONE)

*******************************************************************/
*               CHANNEL DEFINITIONS over TCP                      */
* start runmqchl -c OSBRTX3T -m OSBR                              */
* start runmqlsr -t tcp -m OSBR                                   */
*******************************************************************/

* OSBR TO QMX3 pair*

    DEFINE CHANNEL('OSBRTX3T') CHLTYPE(SDR)                +
       TRPTYPE(TCP) REPLACE DESCR(' ')          +
       BATCHSZ(4) DISCINT(6000)                 +
       CONNAME('9.24.65.2(1488)') XMITQ('QMX3') +
       SHORTTMR(60) SHORTRTY(10)                +
       LONGTMR(1200) LONGRTY(999999999)               +
       SEQWRAP(999999999) MAXMSGL(4194304)            +
       CONVERT(NO)

    DEFINE CHANNEL('X3TOSBRT') CHLTYPE(RCVR)          +
       TRPTYPE(TCP) REPLACE DESCR(' ')          +
       BATCHSZ(50)                        +
       PUTAUT(DEF) SEQWRAP(999999999)           +
       MAXMSGL(4194304)

*******************************************************************/
*              Define local XMITQ queue                    */
*******************************************************************/

    DEFINE QLOCAL('QMX3') REPLACE               +
       DESCR('messages to QMX3')          +
       DEFPSIST(YES)                +
       MAXDEPTH(640000)                   +
       MAXMSGL(4194304)                   +
       SHARE                        +
       USAGE(XMITQ)                 +
       INITQ(SYSTEM.CHANNEL.INITQ) TRIGGER      +
       TRIGTYPE(FIRST)                    +
       TRIGDATA('OSBRTX3T')

******************************************************************/
* >>>>>>>>>>>>                                  <<<<<<<<<<<<<<<<< */
* >>>>>>>      END OF SETUP: MQEI - AIX TO OSBR           <<<<<<< */
* >>>>>>>>>>>>                                  <<<<<<<<<<<<<<<<< */
******************************************************************/
```

<u>Here are the MQSeries Definitions for MQLSX on QMD2 Queue Manager for Solaris defined on OS/2:</u>

```
******************************************************************/
*             OS/2 BRIDGE DEFINITIONS FOR CPIT LAB                */
*             Notes Domino to OS/2 Host server                    */
*               Creation date :  10-28-1998                       */
*               Change History:  11-20-1998                       */
*               Author:                                           */
******************************************************************/
*   (C) Copyright International Business Machines Corp. 1997,1998  */
*;  All Rights Reserved                                           */
*;  US Government Users Restricted Rights - Use, duplication or    */
*;  disclosure restricted by GSA ADP Schedule Contract with IBM    */
*;  Corp.                                                         */
*;                                                                */
*   Licensed Materials - Property of IBM                          */
*;                                                                */
*;                                                                */
*;     NOTICE TO USERS OF THE SOURCE CODE EXAMPLES             */
*;                                                                */
*; INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THE SOURCE */
*; CODE EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, "AS */
*; IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED,  */
*; INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF        */
*; MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.  THE      */
*; ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE SOURCE CODE*/
*; EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, IS WITH  */
*; YOU.  SHOULD ANY PART OF THE SOURCE CODE EXAMPLES PROVE        */
*; DEFECTIVE, YOU (AND NOT IBM OR AN AUTHORIZED DEALER) ASSUME THE */
*; ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.   */
*;                                                                */
*;                                                                */
*                                                                 */
******************************************************************/
******************************************************************/
*              HUB FOR OS/2 MQLSX: SUN QMGR AND OS/2 BRIDGE       */
*                                                                 */
* Remote Queue Manager at Solaris: QMD2                           */
* Local Queue Manager at OS/2 Bridge: OSBR                        */
******************************************************************/

******************************************************************/

******************************************************************/
*                    DEFINE LOCAL QUEUE                           */
******************************************************************/
```

```
    DEFINE QLOCAL('OS.DLQ') REPLACE       +
       DESCR('Dead Letter Queue')      +
       DEFPSIST(YES)             +
       MAXDEPTH(640000)                +
       MAXMSGL(4194304)                +
       SHARE                     +
       USAGE(NORMAL)             +
       TRIGTYPE(NONE)

    DEFINE QLOCAL('LEGACY.LOCAL') REPLACE +
       DESCR('INPUT QUEUE')            +
       DEFPSIST(YES)             +
       MAXDEPTH(640000)                +
       MAXMSGL(4194304)                +
       SHARE                     +
       USAGE(NORMAL)             +
       TRIGTYPE(NONE)


    DEFINE QLOCAL('CLIENT.LOCAL') REPLACE +
       DESCR('OUTPUT Queue')           +
       DEFPSIST(YES)             +
       MAXDEPTH(640000)                +
       MAXMSGL(4194304)                +
       SHARE                     +
       USAGE(NORMAL)             +
       TRIGTYPE(NONE)

  DEFINE QLOCAL('SYSTEM.CICS.BRIDGE.QUEUE') REPLACE   +
       DESCR('MQ general message queue')        +
       DEFPSIST(YES)                      +
       MAXDEPTH(640000)                         +
       MAXMSGL(4194304)                         +
       SHARE                              +
       USAGE(NORMAL)                      +
       TRIGTYPE(NONE)

********************************************************************/
*              RESPONSE QUEUE                               */
********************************************************************/

*  QMD2 REMOTE QUEUE *

   DEFINE QREMOTE('TOD2MQQ') REPLACE   +
         DESCR('messages to QMD2')     +
         PUT(ENABLED)                  +
         DEFPRTY(0)                    +
         DEFPSIST(YES)                 +
         XMITQ('QMD2')           +
         RNAME('D2MQQ')          +
         RQMNAME('QMD2')         +
         SCOPE(QMGR)

********************************************************************/
*               CHANNEL DEFINITIONS over TCP                          */
* start runmqchl -c OSBRTD2T -m OSBR                               */
* start runmqlsr -t tcp -m OSBR                                    */
********************************************************************/
```

```
* OSBR and QMD2 pair *

    DEFINE CHANNEL('OSBRTD2T') CHLTYPE(SDR)                 +
       TRPTYPE(TCP) REPLACE DESCR(' ')                  +
       BATCHSZ(4) DISCINT(6000)                   +
       CONNAME('9.24.xx.xx(1485)') XMITQ('QMD2')        +
       SHORTTMR(60) SHORTRTY(10)                  +
       LONGTMR(1200) LONGRTY(999999999)           +
       SEQWRAP(999999999) MAXMSGL(4194304)              +
       CONVERT(YES)

    DEFINE CHANNEL('D2TOSBRT') CHLTYPE(RCVR)      +
       TRPTYPE(TCP) REPLACE DESCR(' ')            +
       BATCHSZ(50)                       +
       PUTAUT(DEF) SEQWRAP(999999999)       +
       MAXMSGL(4194304)

*******************************************************************/
*                DEFINE LOCAL "XMITQ" QUEUE                        */
*******************************************************************/

    DEFINE QLOCAL('QMD2') REPLACE                 +
       DESCR('messages to QMD2')            +
       DEFPSIST(YES)                     +
       MAXDEPTH(640000)                     +
       MAXMSGL(4194304)                     +
       SHARE                             +
       USAGE(XMITQ)                      +
       INITQ(SYSTEM.CHANNEL.INITQ) TRIGGER        +
       TRIGTYPE(FIRST)                      +
       TRIGDATA('OSBRTD2T')

*******************************************************************/
* >>>>>>>>>>>>                                 <<<<<<<<<<<<<<<<< */
* >>>>>>>     END OF SETUP: MQLSX - DALLAS TO OSBR        <<<<<<< */
* >>>>>>>>>>>>                                 <<<<<<<<<<<<<<<<< */
*******************************************************************/
*******************************************************************/
```

<u>Here are the MQSeries Definitions for MQEI on QMD3 Queue Manager for Solaris defined on OS/2:</u>

```
*******************************************************************/
*                   MQEI: SUN QMGR AND OS/2 BRIDGE                 */
*                                                                  */
*     Queue Managers: Remote (DAL): QMD3 ; Local (OS/2): OSBR      */
*******************************************************************/

*******************************************************************/
*                    DEFINE LOCAL QUEUE                            */
*******************************************************************/

    DEFINE QLOCAL('OS.DLQ') REPLACE      +
       DESCR('Dead Letter Queue')     +
       DEFPSIST(YES)            +
       MAXDEPTH(640000)               +
       MAXMSGL(4194304)                +
       SHARE                     +
       USAGE(NORMAL)             +
       TRIGTYPE(NONE)

    DEFINE QLOCAL('LEGACY.LOCAL') REPLACE +
       DESCR('INPUT QUEUE')            +
       DEFPSIST(YES)             +
       MAXDEPTH(640000)                +
       MAXMSGL(4194304)                +
       SHARE                     +
       USAGE(NORMAL)             +
       TRIGTYPE(NONE)

    DEFINE QLOCAL('CLIENT.LOCAL') REPLACE +
       DESCR('OUTPUT Queue')           +
       DEFPSIST(YES)             +
       MAXDEPTH(640000)                +
       MAXMSGL(4194304)                +
       SHARE                     +
       USAGE(NORMAL)             +
       TRIGTYPE(NONE)

  DEFINE QLOCAL('SYSTEM.CICS.BRIDGE.QUEUE') REPLACE   +
       DESCR('MQ general message queue')        +
       DEFPSIST(YES)                      +
       MAXDEPTH(640000)                         +
       MAXMSGL(4194304)                         +
       SHARE                              +
       USAGE(NORMAL)                      +
       TRIGTYPE(NONE)

*******************************************************************/
*               CHANNEL DEFINITIONS over TCP                      */
* start runmqchl -c OSBRTD3T -m OSBR                              */
* start runmqlsr -t tcp -m OSBR                                   */
*******************************************************************/

* OSBR TO QMD3 pair*

    DEFINE CHANNEL('OSBRTD3T') CHLTYPE(SDR)                +
       TRPTYPE(TCP) REPLACE DESCR(' ')          +
       BATCHSZ(4) DISCINT(6000)                 +
       CONNAME('9.24.xx.xx(1486)') XMITQ('MQD3')      +
       SHORTTMR(60) SHORTRTY(10)                +
       LONGTMR(1200) LONGRTY(999999999)               +
       SEQWRAP(999999999) MAXMSGL(4194304)            +
       CONVERT(NO)

    DEFINE CHANNEL('D3TOSBRT') CHLTYPE(RCVR)    +
       TRPTYPE(TCP) REPLACE DESCR(' ')    +
       BATCHSZ(50)                     +
       PUTAUT(DEF) SEQWRAP(999999999)     +
       MAXMSGL(4194304)

*******************************************************************/
*               DEFINE LOCAL "XMITQ" QUEUE                        */
*******************************************************************/
```

```
    DEFINE QLOCAL('MQD3') REPLACE               +
       DESCR('messages to MQD3')          +
       DEFPSIST(YES)                   +
       MAXDEPTH(640000)                   +
       MAXMSGL(4194304)                   +
       SHARE                           +
       USAGE(XMITQ)                    +
       INITQ(SYSTEM.CHANNEL.INITQ) TRIGGER      +
       TRIGTYPE(FIRST)                    +
       TRIGDATA('OSBRTD3T')

********************************************************************/
* >>>>>>>>>>>>                                   <<<<<<<<<<<<<<<<< */
* >>>>>>>     END OF SETUP: MQEI - DALLAS TO OSBR          <<<<<<< */
* >>>>>>>>>>>>                                   <<<<<<<<<<<<<<<<< */
********************************************************************/
```

<u>Here are the MQSeries Definitions for MQLSX on QMP2 Queue Manager for OS390 defined on OS/2:</u>

```
********************************************************************/
*            OS/2 BRIDGE DEFINITIONS FOR CPIT LAB                  */
*            Notes Domino to OS/2 Host server                      */
*              Creation date :  10-28-1998                         */
*              Change History:  11-20-1998                         */
*                                                                  */
********************************************************************/
*   (C) Copyright International Business Machines Corp. 1997,1998  */
*;  All Rights Reserved                                            */
*;  US Government Users Restricted Rights - Use, duplication or    */
*;  disclosure restricted by GSA ADP Schedule Contract with IBM    */
*;  Corp.                                                          */
*;                                                                 */
*   Licensed Materials - Property of IBM                           */
*;                                                                 */
*;                                                                 */
*;     NOTICE TO USERS OF THE SOURCE CODE EXAMPLES             */
*;                                                                 */
*; INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THE SOURCE */
*; CODE EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, "AS */
*; IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED,  */
*; INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF         */
*; MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.  THE      */
*; ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE SOURCE CODE*/
*; EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, IS WITH  */
*; YOU.  SHOULD ANY PART OF THE SOURCE CODE EXAMPLES PROVE         */
*; DEFECTIVE, YOU (AND NOT IBM OR AN AUTHORIZED DEALER) ASSUME THE */
*; ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.   */
*;                                                                 */
*;                                                                 */
*                                                                  */
********************************************************************/
********************************************************************/
*             HUB FOR OS/2 MQLSX: OS390 QMGR AND OS/2 BRIDGE       */
*                                                                  */
* Remote Queue Manager at OS390: QMP2                              */
* Local Queue Manager at OS/2 Bridge: OSBR                         */
```

```
*******************************************************************/

*******************************************************************/

*******************************************************************/
*                    DEFINE LOCAL QUEUE                           */
*******************************************************************/

    DEFINE QLOCAL('OS.DLQ') REPLACE       +
       DESCR('Dead Letter Queue')      +
       DEFPSIST(YES)             +
       MAXDEPTH(640000)                +
       MAXMSGL(4194304)                +
       SHARE                     +
       USAGE(NORMAL)             +
       TRIGTYPE(NONE)

    DEFINE QLOCAL('LEGACY.LOCAL') REPLACE +
       DESCR('INPUT QUEUE')            +
       DEFPSIST(YES)             +
       MAXDEPTH(640000)                +
       MAXMSGL(4194304)                +
       SHARE                     +
       USAGE(NORMAL)             +
       TRIGTYPE(NONE)


    DEFINE QLOCAL('CLIENT.LOCAL') REPLACE +
       DESCR('OUTPUT Queue')           +
       DEFPSIST(YES)             +
       MAXDEPTH(640000)                +
       MAXMSGL(4194304)                +
       SHARE                     +
       USAGE(NORMAL)             +
       TRIGTYPE(NONE)

  DEFINE QLOCAL('SYSTEM.CICS.BRIDGE.QUEUE') REPLACE   +
       DESCR('MQ general message queue')        +
       DEFPSIST(YES)                      +
       MAXDEPTH(640000)                         +
       MAXMSGL(4194304)                         +
       SHARE                              +
       USAGE(NORMAL)                      +
       TRIGTYPE(NONE)

*******************************************************************/
*                   RESPONSE QUEUE                            */
*******************************************************************/

* QMP2 REMOTE QUEUE *

   DEFINE QREMOTE('TOP2MQQ') REPLACE   +
         DESCR('messages to QMP2')     +
         PUT(ENABLED)                  +
         DEFPRTY(0)            +
         DEFPSIST(YES)                 +
         XMITQ('QMP2')                 +
         RNAME('P2MQQ')                +
         RQMNAME('QMP2')       +
         SCOPE(QMGR)

*******************************************************************/
*               CHANNEL DEFINITIONS over TCP                      */
* start runmqchl -c OSBRTP2T -m OSBR                              */
* start runmqlsr -t tcp -m OSBR                                   */
*******************************************************************/

* OSBR AND QMP2 pair *

   DEFINE CHANNEL('OSBRTP2T') CHLTYPE(SDR)            +
         TRPTYPE(TCP) REPLACE DESCR(' ')        +
         BATCHSZ(4) DISCINT(6000)               +
         CONNAME('9.24.66.42(1481)') XMITQ('QMP2')    +
         SHORTTMR(60) SHORTRTY(10)              +
         LONGTMR(1200) LONGRTY(999999999)       +
         SEQWRAP(999999999) MAXMSGL(4194304)          +
         CONVERT(YES)

   DEFINE CHANNEL('P2TOSBRT') CHLTYPE(RCVR)           +
         TRPTYPE(TCP) REPLACE DESCR(' ')        +
         BATCHSZ(50)                      +
         PUTAUT(DEF) SEQWRAP(999999999)         +
         MAXMSGL(4194304)

*******************************************************************/
*              DEFINE LOCAL "XMITQ" QUEUE                */
*******************************************************************/

   DEFINE QLOCAL('QMP2') REPLACE          +
         DESCR('messages to QMP2')        +
         DEFPSIST(YES)                    +
         MAXDEPTH(640000)              +
         MAXMSGL(4194304)              +
         SHARE                         +
         USAGE(XMITQ)                     +
         INITQ(SYSTEM.CHANNEL.INITQ) TRIGGER    +
         TRIGTYPE(FIRST)               +
         TRIGDATA('OSBRTP2T')

*******************************************************************/
* >>>>>>>>>>>>                                 <<<<<<<<<<<<<<<<< */
* >>>>>>>         END OF SETUP: MQLSX - QMP2             <<<<<<< */
* >>>>>>>>>>>>                                 <<<<<<<<<<<<<<<<< */
*******************************************************************/
```

In the following, CSQP is our QMGR.

```
* (C) Copyright International Business Machines Corp. 1997,1998
*  All Rights Reserved
*  US Government Users Restricted Rights - Use, duplication or
*  disclosure restricted by GSA ADP Schedule Contract with IBM
*  Corp.
*
```

```
*  Licensed Materials - Property of IBM
*
*
*      NOTICE TO USERS OF THE SOURCE CODE EXAMPLES
*
* INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THE SOURCE
* CODE EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS,
* "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR
* IMPLIED, INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF
* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.  THE
* ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE SOURCE CODE
* EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, IS WITH
* YOU.  SHOULD ANY PART OF THE SOURCE CODE EXAMPLES PROVE
* DEFECTIVE, YOU (AND NOT IBM OR AN AUTHORIZED DEALER) ASSUME THE
* ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
*
 DEFINE QLOCAL('SYSTEM.CICS.BRIDGE.QUEUE') REPLACE +
        DESCR('CICS BRIDGE REQUEST QUEUE') +
*                                          Permit shared access
         SHARE +
*                                          FIFO Delivery
         MSGDLVSQ(FIFO) +
*                                          Persistent
         DEFPSIST(YES) +
*                                          Backout hardened
         HARDENBO +
*                                          Trigger on 1st message
*                                          using default initq
         TRIGGER TRIGTYPE(FIRST) +
         PROCESS('CICS_BRIDGE') +
         INITQ('CICS01.INITQ')
```

Notes :
CICS01.INITQ is the name of the Initiation queue we specify when we start the MQSeries-CICS Adapter
The definition for CICS01.INITQ comes with MQ.

```
****************************************************************************

** CSQPMQQ is a Local Queue defined to test the MQSeries-CICS/ESA         ***
**  Bridge locally in Poughkeepsie. This was our local "reply-to-queue".  ***
**                                                                        ***
** (C) Copyright International Business Machines Corp. 1997,1998         ***
**  All Rights Reserved                                                   ***
**  US Government Users Restricted Rights - Use, duplication or          ***
**  disclosure restricted by GSA ADP Schedule Contract with IBM Corp.     ***

**                                                                        ***
**  Licensed Materials - Property of IBM                                  ***

**                                                                        ***
**                                                                        ***
**     NOTICE TO USERS OF THE SOURCE CODE EXAMPLES                   ***
**                                                                        ***
** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THE SOURCE CODE   ***
** EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, "AS IS" WITHOUT ***
** WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT  ***
** LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A ***
** PARTICULAR PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE ***
** OF THE SOURCE CODE EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE      ***
** GROUPS, IS WITH YOU.  SHOULD ANY PART OF THE SOURCE CODE EXAMPLES      ***
** PROVE DEFECTIVE, YOU (AND NOT IBM OR AN AUTHORIZED DEALER) ASSUME THE  ***

** ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.          ***
**                                                                        ***
*****************************************************************************
 DEFINE QLOCAL( 'CSQPMQQ' )  REPLACE +

 * Common queue attributes
         DESCR( 'MQ traffic testing' ) +
         PUT( ENABLED ) +
         DEFPRTY( 0 ) +
         DEFPSIST( NO ) +

 * Local queue attributes
         GET( ENABLED ) +
                    NOSHARE +

                    DEFSOPT( EXCL ) +

                    MSGDLVSQ( PRIORITY ) +

                    RETINTVL( 999999999 ) +

                    MAXDEPTH( 999999999 ) +


              MAXMSGL( 4194304 ) +
                    NOHARDENBO +
              BOTHRESH( 0 ) +
                    BOQNAME( ' ' ) +
                    STGCLASS( 'DEFAULT' ) +
                    USAGE( NORMAL ) +

        * Event control attributes
                    QDPMAXEV( ENABLED ) +
                    QDPHIEV( DISABLED ) +
                    QDEPTHHI( 80 ) +
                    QDPLOEV( DISABLED ) +
                    QDEPTHLO( 40 ) +
                    QSVCIEV( NONE ) +
                    QSVCINT( 999999999 ) +

         * Trigger attributes
              NOTRIGGER +
                    TRIGTYPE( FIRST ) +
        TRIGDPTH( 1 ) +
              TRIGMPRI( 0 ) +
              TRIGDATA( ' ' ) +
                    PROCESS( ' ' ) +
              INITQ( ' ' )


 *****************************************************************************
***************************
```

```
 * SENDING END DEFINITIONS FOR CSQP (POUGHKEEPSIE OS/390) TO AUSTIN NT

*******************************************************************************
************************
 *
 ******

 DEFINE QLOCAL( 'QMA1' ) REPLACE +

 * Common queue attributes
          DESCR( 'Transmission queue for QMA1' ) +
          PUT( ENABLED ) +
          DEFPRTY( 5 ) +
          DEFPSIST( YES ) +

* Local queue attributes
         GET( ENABLED ) +
         SHARE +
         DEFSOPT( EXCL ) +
         MSGDLVSQ( FIFO ) +
         RETINTVL( 999999999 ) +
         MAXDEPTH( 10000 ) +
         MAXMSGL( 4194304 ) +
         NOHARDENBO +
         BOTHRESH( 0 ) +
         BOQNAME( ' ' ) +
         STGCLASS( 'REMOTE' ) +
         USAGE( XMITQ ) +

* Event control attributes
         QDPMAXEV( ENABLED ) +
         QDPHIEV( DISABLED ) +
         QDEPTHHI( 80 ) +
         QDPLOEV( DISABLED ) +
         QDEPTHLO( 40 ) +
         QSVCIEV( NONE ) +
         QSVCINT( 999999999 ) +

* Trigger attributes
         TRIGGER +
         TRIGTYPE( FIRST ) +
         TRIGMPRI( 0 ) +
         TRIGDPTH( 1 ) +
         TRIGDATA( 'CSQPTA1T' ) +
         PROCESS( 'QMA1.SEND.PROCESS' ) +
         INITQ( 'SYSTEM.CHANNEL.INITQ' )
*
******
DEFINE PROCESS( 'QMA1.SEND.PROCESS' ) REPLACE +

* Process attributes
         DESCR( 'Process for sending messages to QMA1' ) +
         APPLTYPE( MVS ) +
         APPLICID( 'CSQX START' ) +
         USERDATA( 'CSQPTA1T' ) +
         ENVRDATA( ' ' )
*

DEFINE CHANNEL( 'CSQPTA1T') +
         CHLTYPE( SDR ) +

* Sender Channel Attributes

         DESCR( 'Channel for sending messages to QMA1' ) +
         TRPTYPE( TCP ) REPLACE +
         XMITQ( 'QMA1' ) +
         MCAUSER( ' ' ) +
         BATCHSZ( 50 ) +
         DISCINT( 0 ) +
         SHORTRTY( 10 )       SHORTTMR( 60 ) +
         LONGRTY( 999999999 ) LONGTMR( 120 ) +
         SCYEXIT( ' ' )       SCYDATA( ' ' ) +
         MSGEXIT( ' ' )       MSGDATA( ' ' ) +
         SENDEXIT( ' ' )      SENDDATA( ' ' ) +
         RCVEXIT( ' ' )       RCVDATA( ' ' ) +
         SEQWRAP( 999999999 ) +
         CONVERT( YES ) +
         MAXMSGL( 4194304 ) +

* This defines where the target queue manager is.
* Specify either the TCP/IP network address, for example
         CONNAME( '9.67.142.85' )      VERIFY/CHANGE IP ADDR

* or its TCP/IP host name provided by the name server, for example
*        CONNAME( 'QMA1' )

*
****** TO AUSTIN #1
**         server to server use TOA1MQQ reply to queue     ****

DEFINE QREMOTE( 'TOA1MQQ' ) REPLACE +

* Common queue attributes
         DESCR( 'Queue for accessing MQQ on QMA1' ) +
         PUT( ENABLED ) +
         DEFPSIST( YES ) +
         DEFPRTY( 9 ) +

* Remote queue attributes
*  RNAME IS QUEUE ON AUSTIN
*  RQMNAME IS QUEUE MANAGER ON AUSTIN
         RNAME( 'A1MQQ' ) +
         RQMNAME( QMA1 ) +
         XMITQ( 'QMA1' )
*

* A1MQQ also needs to be defined as a LOCAL queue on QMA1.
*
*
******************************************************************************
*********************************
* RECEIVING END DEFINITIONS FOR CSQP (POUGHKEEPSIE OS/390) FROM AUSTIN NT

******************************************************************************
*********************************
*
*
DEFINE CHANNEL( 'A1TCSQPT') +
         CHLTYPE( RCVR ) +

* Receiver channel attributes
         DESCR( 'Channel for receiving messages from QMA1' ) +
         TRPTYPE( TCP ) REPLACE +
         BATCHSZ( 50 ) +
         SCYEXIT( ' ' )       SCYDATA( ' ' ) +
         MSGEXIT( ' ' )       MSGDATA( ' ' ) +
         SENDEXIT( ' ' )      SENDDATA( ' ' ) +
         RCVEXIT( ' ' )       RCVDATA( ' ' ) +
         MCAUSER( ' ' ) +
         PUTAUT( DEF ) +
         SEQWRAP( 999999999 ) +
         MAXMSGL( 4194304 )

*
*****************************************************************
* End of CSQPTA1T
*****************************************************************

******************************************************************************
*********************
* SENDING END DEFS FOR CSQP (POUGHKEEPSIE OS/390) TO AUSTIN OS/2
******************************************************************************
*********************
*
******
DEFINE QLOCAL( 'QMO1' ) REPLACE +

* Common queue attributes
         DESCR( 'Transmission queue for QMO1' ) +
         PUT( ENABLED ) +
         DEFPRTY( 5 ) +
         DEFPSIST( YES ) +

* Local queue attributes
         GET( ENABLED ) +
         SHARE +
         DEFSOPT( EXCL ) +
         MSGDLVSQ( FIFO ) +
         RETINTVL( 999999999 ) +
         MAXDEPTH( 10000 ) +
         MAXMSGL( 4194304 ) +
         NOHARDENBO +
         BOTHRESH( 0 ) +
         BOQNAME( ' ' ) +
         STGCLASS( 'REMOTE' ) +
         USAGE( XMITQ ) +

* Event control attributes
         QDPMAXEV( ENABLED ) +
         QDPHIEV( DISABLED ) +
         QDEPTHHI( 80 ) +
         QDPLOEV( DISABLED ) +
         QDEPTHLO( 40 ) +
         QSVCIEV( NONE ) +
         QSVCINT( 999999999 ) +

* Trigger attributes
```

```
         TRIGGER +
         TRIGTYPE( FIRST ) +
         TRIGMPRI( 0 ) +
         TRIGDPTH( 1 ) +
         TRIGDATA( 'CSQPTO1T' ) +
         PROCESS( 'QMO1.SEND.PROCESS' ) +
         INITQ( 'SYSTEM.CHANNEL.INITQ' )
*
******
DEFINE PROCESS( 'QMO1.SEND.PROCESS' ) REPLACE +

* Process attributes
         DESCR( 'Process for sending messages to QMO1' ) +
         APPLTYPE( MVS ) +
         APPLICID( 'CSQX START' ) +
         USERDATA( 'CSQPTO1T' ) +
         ENVRDATA( ' ' )
*
****** CSQP (ME) TO AUSTIN #2 USING TCPIP
DEFINE CHANNEL( 'CSQPTO1T') +
         CHLTYPE( SDR ) +

* Sender channel attributes
         DESCR( 'Channel for sending messages to QMO1' ) +
         TRPTYPE( TCP ) REPLACE +
         XMITQ( 'QMO1' ) +
         MCAUSER( ' ' ) +
         BATCHSZ( 50 ) +
         DISCINT( 0 ) +
         SHORTRTY( 10 )       SHORTTMR( 60 ) +
         LONGRTY( 999999999 ) LONGTMR( 120 ) +
         SCYEXIT( ' ' )       SCYDATA( ' ' ) +
         MSGEXIT( ' ' )       MSGDATA( ' ' ) +
         SENDEXIT( ' ' )      SENDDATA( ' ' ) +
         RCVEXIT( ' ' )       RCVDATA( ' ' ) +
         SEQWRAP( 999999999 ) +
         CONVERT( YES ) +
         MAXMSGL( 4194304 ) +

* This defines where the target queue manager is.
* Specify either the TCP/IP network address, for example
         CONNAME( '9.67.142.86' )    VERIFY/CHANGE IP ADDR
* or its TCP/IP host name provided by the name server, for example
*        CONNAME( 'QMO1' )
*
**         server to server use TOO1MQQ reply to queue     ****
DEFINE QREMOTE( 'TOO1MQQ' ) REPLACE +

* Common queue attributes
         DESCR( 'Queue for accessing MQQ on QMO1' ) +
         PUT( ENABLED ) +
         DEFPSIST( YES ) +
         DEFPRTY( 9 ) +

* Remote queue attributes
*  RNAME IS QUEUE ON AUSTIN
*  RQMNAME IS QUEUE MANAGER ON AUSTIN
         RNAME( 'O1MQQ' ) +
         RQMNAME( QMO1 ) +
         XMITQ( 'QMO1' )
*
* O1MQQ also needs to be defined as a LOCAL queue on QMO1.
*
*
******************************************************************************
*******
* RECEIVING END DEFS FOR CSQP (POUGHKEEPSIE OS/390) FROM
* AUSTIN OS/2
******************************************************************************
*******
*
DEFINE CHANNEL( 'O1TCSQPT') +
         CHLTYPE( RCVR ) +

* Receiver channel attributes
         DESCR( 'Channel for receiving messages from QMO1' ) +
         TRPTYPE( TCP ) REPLACE +
         BATCHSZ( 50 ) +
         SCYEXIT( ' ' )       SCYDATA( ' ' ) +
         MSGEXIT( ' ' )       MSGDATA( ' ' ) +
         SENDEXIT( ' ' )      SENDDATA( ' ' ) +
         RCVEXIT( ' ' )       RCVDATA( ' ' ) +
         MCAUSER( ' ' ) +
         PUTAUT( DEF ) +
         SEQWRAP( 999999999 ) +
         MAXMSGL( 4194304 )
*
*****************************************************************
* End of CSQPTO1T
*****************************************************************
```

Appendix C - CPIT MQSeries-CICS Bridge

The CPIT MQSeries-CICS Bridge is a background running job in a CICS region. This bridge checks the MQ input queue every one second. If it finds a message in the input queue, it will extract the message and link to the program which is in the message. After returning back from the linked program, the bridge will put a message back to the reply queue from COMMAREA. The name of the input message queue is hard coded in the bridge but the reply queue name is taken from the header of the message, MQMD. This bridge works for Lotus MQLSX and MQEI. When MsgDesc.Format is equal to 'MQCICS', this message is from MQEI and the linking program name is on location 165 to 172 of the input message. If the format is not 'MQCICS', this message is from MQLSX and the linking program name is on location 1 to 8 of the input message.

The CPIT MQSeries-CICS bridge can be started by running transaction 'MCST' which will run 'MCLP' in the background. This bridge can be terminated by sending an MQ message with the program name equal to 'EXIT   '. There is a big difference between this bridge and the MQSeries-CICS/ESA Bridge for OS/390. On OS/390, a CICS Bridge Monitor is checking the input message queue. If there is one message in the queue, the monitor will start a CICS bridge task to link to the DPL program. If there are more than one messages, the monitor may start multiple CICS bridge tasks at the same time. On the other hand, the bridge on NT can not start multiple CICS bridge tasks. The CICS region on NT does not connect to MQ when the region is up. Due to no connection to MQ in CICS region on NT, the bridge must issue the connection to MQ every time which will take about 60 seconds. If the same design as OS/390 is used, every bridge task takes 60 seconds to finished its request. The way to solve this problem is to keep the connection alive. Let the bridge connect to MQ once. Retrieve the input message and link to the DPL program one at a time. Then, come back and retrieve the next message in the same bridge task without issuing another MQ connection. This process takes less than one second per message.

CPIT MQSeries-CICS Bridge Source

The source code for **MCST** follows:

```
/*
*
* NAME:          mcstart.sqc     Transaction ID : MCST
* VERSION:       1.0
* COPYRIGHT:
*
*   (C) COPYRIGHT International Business Machines Corp. 1993, 1997
*   All Rights Reserved
*   Licensed Materials - Property of IBM
*
*   US Government Users Restricted Rights - Use, duplication or
*   disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
*
*
*            NOTICE TO USERS OF THE SOURCE CODE EXAMPLES
*
* INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THE SOURCE CODE
* EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, "AS IS"
* WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING,
* BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
* FITNESS FOR A PARTICULAR PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY
* AND PERFORMANCE OF THE SOURCE CODE EXAMPLES, BOTH INDIVIDUALLY AND
* AS ONE OR MORE GROUPS, IS WITH YOU.  SHOULD ANY PART OF THE SOURCE
* CODE EXAMPLES PROVE DEFECTIVE, YOU (AND NOT IBM) ASSUME THE ENTIRE
* COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
*
*
*    Start a cics transaction and display a message on the screen.
*
* Transaction: MCST
* Program:     MCSTART
* Mapset:
*/

#ifdef _WIN32
#define DLLIMPORT __declspec(dllimport)
#define DLLEXPORT __declspec(dllexport)
#define CDECL __cdecl
#else
#define DLLIMPORT
#define DLLEXPORT
#define CDECL
#endif

#include <stdio.h>
#ifdef _WIN32
#include <stdlib.h>
#endif
/*
#include "uxa1.h"
  */
#ifdef _WIN32
#include <string.h>
#endif

#define STARTMSG "Start a MQ monitor transaction."

DLLEXPORT CDECL main()
{
        char errmsg[400];
        char qmsg[400];
   short mlen;


        /*   Get addresability for EIB */

        EXEC CICS ADDRESS EIB(dfheiptr);

   sprintf(qmsg, "%s", "Running Transaction MCLP");
   mlen = strlen(qmsg);

   EXEC CICS SEND FROM (qmsg) LENGTH(mlen) ERASE;

   EXEC CICS START TRANSID("MCLP");
        EXEC CICS RETURN;
}
```

The source code for **MCLP** follows:

```
/*********************************************************************/
/*                                                                   */
/* Program name: MCLOOP   Transaction ID : MCLP                      */
/*                        V. 2.10                                    */
/*                        022599 - Initialized MQOPEN when Client    */
/*                                 was changed.                      */
/*   (C) COPYRIGHT International Business Machines Corp. 1993, 1997 */
/*   All Rights Reserved                                             */
/*   Licensed Materials - Property of IBM                            */
/*                                                                   */
/*   US Government Users Restricted Rights - Use, duplication or     */
/*   disclosure restricted by GSA ADP Schedule Contract with         */
/*   IBM Corp.                                                       */
/*                                                                   */
/*                                                                   */
/*            NOTICE TO USERS OF THE SOURCE CODE EXAMPLES            */
/*                                                                   */
/* INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THE SOURCE   */
/* CODE EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, "AS   */
/* IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED,    */
/* INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF           */
/* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.  THE        */
/* ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE SOURCE CODE */
/* EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, IS WITH    */
/* YOU.  SHOULD ANY PART OF THE SOURCE CODE EXAMPLES PROVE           */
/* DEFECTIVE, YOU (AND NOT IBM) ASSUME THE ENTIRE COST OF ALL        */
/* NECESSARY SERVICING, REPAIR OR CORRECTION.                        */
/*                                                                   */
/*                                                                   */
/* Environment :                                                     */
/*              Transaction Server for Windows NT Version 4.0        */
/*                                                                   */
/* Description :                                                     */
/*              The MQ connection and open are included in this      */
/*               program and keeps outside the loop of mqget.        */
/*              This transaction gets the message from message       */
/*              queue and links to the PROGRAM which is the          */
/*              first 8 characters on the message.   If the          */
/*              PROGRAM ID is invalid, an error message will         */
/*              be sent back.                                        */
/*                                                                   */
/*  Statement:     Licensed Materials - Property of IBM              */
/*                                                                   */
/*                    Program logic                                  */
/*                    -------------                                  */
/*                                                                   */
/*  main                                                             */
/*  ----                                                             */
/*       Establish CICS addressability                               */
/*       Connect to default Queue Manager                            */
/*       If connect unsuccessful                                     */
/*          Call RecordCallError                                     */
/*       Else                                                        */
```

```
 /*         Open the input queue SYSTEM.CICS.BRIDGE.QUEUE           */
/*           Open the output queue CLIENT.LOCAL.  If ReplyToQ is used,
                       PUT the message to ReplyToQ.                 */
 /*         If open unsuccessful                                     */
 /*            Call RecordCallError                                  */
 /*         Else                                                     */
 /*            Set the MCGET call parameters                         */
 /*            Get a message from the input queue with wait          */
 /*            Do while there are messages to be retrieved           */
 /*                  Call ProcessMessage                             */
 /*               Reset the call parameters                          */
 /*               Get next message from the input queue with wait    */
 /*            End-do                                                */
 /*            If loop has ended for reason other than 'no           */
 /*               message available'                                 */
 /*               Call RecordCallError                               */
 /*            End-if                                                */
 /*            Close the input queue                                 */
 /*            If close unsuccessful                                 */
 /*               Call RecordCallError                               */
 /*            End-if                                                */
 /*         End-if                                                   */
 /*         Disconnect from the Queue Manager                        */
 /*       End-if                                                     */
 /*                                                                  */
 /*       Return to CICS                                             */
 /*                                                                  */
 /*  ProcessMessage                                                  */
 /*  --------------                                                  */
 /*       Link  a new transaction and get the data back.             */
 /*                                                                  */
 /*       Return to main function                                    */
 /*                                                                  */
 /*                                                                  */
 /*  CheckQUnknownMsg                                                */
 /*  ----------------                                                */
 /*       Call RecordCallError                                       */
 /*       Call ForwardMsgToDLQ, to forward the                       */
 /*       message to the sample's dead-letter queue                  */
 /*                                                                  */
 /*       Return to main function                                    */
 /*                                                                  */
 /*                                                                  */
 /*  RecordCallError                                                 */
 /*  ---------------                                                 */
 /*       Get the time from CICS                                     */
 /*       Get CICS to format the date and time                       */
 /*       Call PackToLong to convert the packed decimal              */
 /*       number to an unpacked number                               */
 /*       Build the output message data                              */
 /*       Write the call error message to CSMT TDQ                   */
 /*       Write the log error message to CSMT TDQ                    */
 /*                                                                  */
 /*       Return to main function                                    */
 /*                                                                  */
 /*                                                                  */
 /* ForwardMsgToDLQ                                                  */
 /* ---------------                                                  */
 /*      Set the MQPUT1 call parameters to enable the message        */
 /*      to be put on the samples dead-letter queue                  */
 /*      If the message to be sent is longer than the buffer         */
 /*         Set message length to the full buffer length             */
 /*      End-if                                                      */
 /*      Put the message on the sample's dead-letter queue           */
 /*         Call RecordCallError                                     */
 /*                                                                  */
 /*      Return to main function                                     */
 /*                                                                  */
 /*                                                                  */
 /* PackToLong                                                       */
 /* ----------                                                       */
 /*      Convert the CICS packed decimal task number to a            */
 /*      long integer                                                */
 /*                                                                  */
 /*      Return to main function                                     */
 /*                                                                  */
 /*                                                                  */
 /********************************************************************/

#include <stdio.h>                          /* used for standard I/O */
#include <string.h>                         /* string functions      */
#include <stdlib.h>
#include <time.h>                           /* time functions        */
#include <dfhaid.h>                         /* CICS header file      */
#include <cmqc.h>                           /* MQI header file       */
#include "monitor.h"                        /* Local definitions     */
/*  a section for CPIT  */
char     qmsg[70];
char     tran_id[5];
char     prog_name[9];
int      getct;
char     addr1 [8];
char    mqdata[70];

static short int  comm_length = 70;
static char  *commarea;
int   rcode;

MQLONG   Hconn;                          /* Connection handle        */
MQHOBJ   Hobj_InputQ;                    /* Object handle            */

/*  SAVE the hconn and qhandle for mqget */
MQLONG   g_hconn;
MQHOBJ   g_handle;
/*  SAVE the hconn and qhandle for mqput */
MQLONG   p_hconn;
MQHOBJ   p_handle;

MQLONG   CompCode;                       /* Completion code          */
MQLONG   Reason;                         /* Qualifying reason        */
               /* c:\mqm\Tools\c\include\cmqc.h            */
MQOD     ObjDesc    = {MQOD_DEFAULT};    /* Object descriptor        */
MQMD     MsgDesc    = {MQMD_DEFAULT};    /* Message descriptor       */
MQLONG   OpenOptions;                    /* Control the MQOPEN call  */

MQGMO    GetMsgOpts = {MQGMO_DEFAULT};   /* Get-Message Options      */

/* MQLONG   MsgBuffLen;  */              /* Length of message buffer */
/* MONITOR_INPUT MsgBuffer;  */          /* Message structure        */
int      MsgBuffLen;
char     MsgBuffer[400];
char     PutBuffer[400];

MQLONG         DataLen;                  /* Length of message        */

MONITOR_DLQ   DLQBuffer;                /* Message structure        */
MQLONG         DLQLen;                   /* Length of message        */

MQPMO    PutMsgOpts = {MQPMO_DEFAULT};   /* Put-Message Options      */
MQLONG   PutBuffLen;                     /* Length of message buffer */

MQCHAR48 QMName = "";                    /* Queue Manager Name       */
MQCHAR48 InputQName = "SYSTEM.CICS.BRIDGE.QUEUE";   /* Input Queue Name */
MQLONG   InputLength = MQ_Q_NAME_LENGTH;
MQCHAR48 PRE_QNAME = "PREVIOUS REPLY TO QUEUE";   /* Previous ReplyToQ Name
*/


char     ErrorMsg[160];         /* Error Message Buffer              */
char     Operation[21];         /* Operation in which error occurred */
char     ObjName[MQ_Q_NAME_LENGTH];               /* Name of Object */


int      bConnected = FALSE;
int      bBadMsg    = FALSE;

/*********************************************************************/
/* Function prototypes                                               */
/*********************************************************************/
void ProcessMessage(void);
void CheckQUnknownMsg(void);
void RecordCallError(void);
void ForwardMsgToDLQ(void);
void PackToLong(unsigned char *source,
                unsigned int source_size,
                long int *target);


/*********************************************************************/
/*  The main function initializes and controls the program flow.     */
/*                                                                   */
/*  After opening the input queue, the program enters a loop getting */
/*  and processing messages.  Once no more messages are available,   */
/*  shown by the wait interval expiring, control is returned to CICS.*/
/*********************************************************************/

void main(void)
{
int    getct;

   strcpy(addr1, "mqdata=");
   MsgBuffLen = 400;
   /******************************************************************/
   /* Establish CICS addressability                                  */
   /******************************************************************/
   EXEC CICS ADDRESS EIB(dfheiptr);

   EXEC CICS ADDRESS COMMAREA(commarea);  /* Get addressability of
                                            the COMMAREA            */
   /*****************************************************************/
   /* Connect to the default Queue Manager                          */
   /*****************************************************************/

   MQCONN(QMName,
          &Hconn,
          &CompCode,
          &Reason);

   /*****************************************************************/
   /* If we failed, then report error and exit                      */
   /*****************************************************************/

   if (CompCode == MQCC_FAILED)
      {
      strncpy(Operation, "MQCONN", sizeof(Operation));
      strncpy(ObjName, QMName, MQ_Q_MGR_NAME_LENGTH);
      RecordCallError();
      }
      else
      {
      bConnected = TRUE;
      }

   /*****************************************************************/
   /* Set up the name of the work queue                             */
   /*****************************************************************/

   strncpy(ObjDesc.ObjectName, InputQName, MQ_Q_NAME_LENGTH);

   /*****************************************************************/
   /* Initialize options and open the queue for input SYSTEM.CICS.BRIDGE.QUEUE
  */
   /*****************************************************************/

   /*  save the Hconn for mqget and mqput  */
    g_hconn = Hconn;
    p_hconn = Hconn;

         OpenOptions = MQOO_INPUT_AS_Q_DEF +
                       MQOO_FAIL_IF_QUIESCING;

         MQOPEN(g_hconn,
                &ObjDesc,
                OpenOptions,
                &g_handle,
                &CompCode,
                &Reason);

/*
         MQOPEN(Hconn,
                &ObjDesc,
                OpenOptions,
                &Hobj_InputQ,
                &CompCode,
                &Reason);
*/

   /****************************************************************/
   /* Test the output from the open, if not OK then build an error   */
   /* message                                                        */
   /****************************************************************/

   if (CompCode == MQCC_FAILED)
      {
      strncpy(Operation, "MQOPEN input", sizeof(Operation));
      strncpy(ObjName, ObjDesc.ObjectName, MQ_Q_NAME_LENGTH);
      RecordCallError();
      }
/*  The ReplyToQ name which comes with the message can be pre-open.  */
  /****************************************************************/
   /* Initialize options and open the client queue for output       */
   /****************************************************************/
/*
   strncpy(ObjDesc.ObjectName, PUTQ_QNAME, MQ_Q_NAME_LENGTH);

         OpenOptions = MQOO_OUTPUT +
                       MQOO_FAIL_IF_QUIESCING;

         MQOPEN(p_hconn,
                &ObjDesc,
                OpenOptions,
                &p_handle,
                &CompCode,
                &Reason);
*/
   /****************************************************************/
   /* Test the output from the open, if not OK then build an error   */
   /* message                                                        */
   /****************************************************************/
/*
   if (CompCode == MQCC_FAILED)
      {
      strncpy(Operation, "MQOPEN output", sizeof(Operation));
      strncpy(ObjName, ObjDesc.ObjectName, MQ_Q_NAME_LENGTH);
      RecordCallError();
      }
*/

      /*************************************************************/
      /* Get and process messages                                    */
      /*************************************************************/

      GetMsgOpts.Options = MQGMO_ACCEPT_TRUNCATED_MSG +
               MQGMO_WAIT;   /* wait for new messages       */
      GetMsgOpts.WaitInterval = 01000; /* 1 second limit for waiting */
      MsgBuffLen = sizeof(MsgBuffer);

      memcpy(MsgDesc.MsgId, MQMI_NONE, sizeof(MsgDesc.MsgId));
      memcpy(MsgDesc.CorrelId, MQCI_NONE, sizeof(MsgDesc.CorrelId));

      /*************************************************************/
      /* Make the first get call outside the loop                    */
      /*************************************************************/
```

```
/*      strcpy( qmsg, "Before Mqget is called");
  EXEC CICS SEND FROM (qmsg) LENGTH(80) ERASE; */

     MQGET(g_hconn,
           g_handle,
           &MsgDesc,
           &GetMsgOpts,
           MsgBuffLen,
           &MsgBuffer,
           &DataLen,
           &CompCode,
           &Reason);

        /**************************************************************/
        /* Handle the message
      Read the queue until 'EXIT' is read.                     */
        /**************************************************************/
   getct = 0;

   for (;;)
   {

/*     sprintf(qmsg, "%s", "After Get a message or wait for message");
     EXEC CICS SEND FROM (qmsg) LENGTH(80) ERASE;
*/
       /* Link to this transaction.  */
       if (Reason == MQRC_NO_MSG_AVAILABLE)
       {
  /* strcpy( qmsg, "After Mqget-no available message");
  EXEC CICS SEND FROM (qmsg) LENGTH(80) ERASE;
  */
  }
  else
  if (CompCode == MQCC_FAILED)
          {
            strncpy(Operation, "MQGET", sizeof(Operation));
            strncpy(ObjName, ObjDesc.ObjectName, MQ_Q_NAME_LENGTH);
            RecordCallError();
      break;
          }
          else
    {

       if (!strncmp(MsgDesc.Format, "MQCICS", 6) )
       {
               memcpy( prog_name, MsgBuffer+164, 4);
            }
       else
       {
               memcpy( prog_name, MsgBuffer, 4);
       }
             /*    strcpy(prog_name+4, "\0");  */
       if (!strncmp(prog_name, "EXIT", 4) )
       {
         sprintf(qmsg, "%s", "EXIT received");
         EXEC CICS SEND FROM (qmsg) LENGTH(80) ERASE;
       /*  ProcessMessage();  */
         break;
       }
```

```
         else
         {

             ProcessMessage();
         }
       }

          /*  jcw - testing program not to exceed 8 times */
    /*
              getct++;
        if (getct >= 8)
        {
          sprintf(qmsg, "%s", "No EXIT received");
            EXEC CICS SEND FROM (qmsg) LENGTH(80) ERASE;
          break;
        }
    */

     /**********************************************************/
     /* Reset parameters for the next get call                 */
     /**********************************************************/

     memcpy(MsgDesc.MsgId, MQMI_NONE, sizeof(MsgDesc.MsgId));
     memcpy(MsgDesc.CorrelId, MQCI_NONE, sizeof(MsgDesc.CorrelId));

       GetMsgOpts.Options = MQGMO_ACCEPT_TRUNCATED_MSG +
                MQGMO_WAIT;   /* wait for new messages       */
       GetMsgOpts.WaitInterval = 01000; /* 1 second limit for waiting */

     MQGET(g_hconn,
           g_handle,
           &MsgDesc,
           &GetMsgOpts,
           MsgBuffLen,
           &MsgBuffer,
           &DataLen,
           &CompCode,
           &Reason);

      } /* end for loop */



 /*  close input queue */

       MQCLOSE(g_hconn,
               &g_handle,
               MQCO_NONE,
               &CompCode,
               &Reason);

       if (CompCode == MQCC_FAILED)
          {
          strncpy(Operation, "MQCLOSE input", sizeof(Operation));
          strncpy(ObjName, InputQName, MQ_Q_NAME_LENGTH);
          RecordCallError();
          }
```

```
   if (bConnected == TRUE)
      {
      MQDISC(&g_hconn,
             &CompCode,
             &Reason);

      if (CompCode == MQCC_FAILED)
         {
         strncpy(Operation, "MQDISC input", sizeof(Operation));
         strncpy(ObjName, QMName, MQ_Q_MGR_NAME_LENGTH);
         RecordCallError();
         }
       }

     /*   close output queue */

    /*  This CONNECT has been closed when close connection for GET */
    /*
      MQCLOSE(p_hconn,
              &p_handle,
              MQCO_NONE,
              &CompCode,
              &Reason);

      if (CompCode == MQCC_FAILED)
         {
         strncpy(Operation, "MQCLOSE output", sizeof(Operation));
         strncpy(ObjName, InputQName, MQ_Q_NAME_LENGTH);
         RecordCallError();
         }



   if (bConnected == TRUE)
      {
      MQDISC(&p_hconn,
             &CompCode,
             &Reason);

      if (CompCode == MQCC_FAILED)
         {
         strncpy(Operation, "MQDISC output", sizeof(Operation));
         strncpy(ObjName, QMName, MQ_Q_MGR_NAME_LENGTH);
         RecordCallError();
         }
      }
     ---------------------------------- */

   /*******************************************************************/
   /* Return to CICS                                                  */
   /*******************************************************************/

   EXEC CICS RETURN;

   } /* end main */


/**********************************************************************/
```

```
/*  This function develops a reply message and puts it on the        */
/*  reply queue, as specified in the input message.  If this         */
/*  operation fails, record the error and forward the message        */
/*  to the sample's dead-letter queue.                               */
/*********************************************************************/

void ProcessMessage(void)
{
short i;
char conv_area [62];    /*  70 bytes - 8 byte program ID = 62 bytes   */
char retn_area [70];
   /*****************************************************************/
   /* Link to this transaction.                                     */
   /*****************************************************************/
/* 022499
   strcpy( qmsg, "Process Message is called");
   EXEC CICS SEND FROM (qmsg) LENGTH(80) ERASE;


   */
         if (!strncmp(MsgDesc.Format, "MQCICS", 6) )
   {
             memcpy( mqdata, MsgBuffer+164, 70);

        }
         else
   {
             memcpy( mqdata, MsgBuffer, 70);
   }
   memcpy(prog_name, mqdata, 8);

   if (!strncmp(prog_name, "EXIT", 4) )
        {
          return;
        }


   memcpy(conv_area, mqdata+8, 62);
   /* commarea is a pointer. Do not use 'memcpy(connarea, conv_area, 70)'   */
   commarea = conv_area;


        EXEC CICS LINK PROGRAM(prog_name) LENGTH(70)
                COMMAREA(commarea) RESP(rcode);

        memcpy(retn_area, commarea, 70);  /* retn_area and commarea must be
*/
                    /*  the same length.                              */

        memcpy(mqdata+8, retn_area, 62);


        if (rcode == DFHRESP(PGMIDERR))
        {
            strcpy(qmsg, "Invalid program name.");
            memcpy(mqdata+8, qmsg, 21);
        }
        else
        {
```

```
        /* memcpy(mqdata, commarea, 70); */
         }

    for (i=0; i<69; i++)
    {
        if (mqdata[i] == '\0')
        {
          mqdata[i] = ' ';
        }
    }

/* JCW  */
/*
    strcpy( qmsg, "??? Hard code ReplyToQ          ");
    EXEC CICS SEND FROM (qmsg) LENGTH(80) ERASE;
 */
    /* --------- This is hard code for testing -----------   */
/*
    strncpy( MsgDesc.ReplyToQ, "TODXMQQ", MQ_Q_NAME_LENGTH);
          strncpy( MsgDesc.ReplyToQMgr, "QMDX", MQ_Q_MGR_NAME_LENGTH);
 */
       /* Put the returned data into Message Queue   */
  /*****************************************************************/
    /* Initialize options and open the ReplyToQ queue for output       */
    /*****************************************************************/

    /*    strncpy(ObjDesc.ObjectName, PUTQ_QNAME, MQ_Q_NAME_LENGTH);   */
    /* ----  Open the ReplyToQ if the queue name is not the same.  ----- */
/* 022499

    strcpy( qmsg, "??? B4 check QNAME-preqn, RepQ " );
    EXEC CICS SEND FROM (qmsg) LENGTH(80) ERASE;


    memcpy( qmsg, PRE_QNAME,sizeof(PRE_QNAME));
    EXEC CICS SEND FROM (qmsg) LENGTH(80) ERASE;


    memcpy( qmsg, MsgDesc.ReplyToQ,sizeof(MsgDesc.ReplyToQ));
    EXEC CICS SEND FROM (qmsg) LENGTH(80) ERASE;


    memcpy( qmsg, MsgDesc.ReplyToQMgr,sizeof(MsgDesc.ReplyToQMgr));
    EXEC CICS SEND FROM (qmsg) LENGTH(80) ERASE;

*/

    if (strncmp(PRE_QNAME, MsgDesc.ReplyToQ, MQ_Q_NAME_LENGTH) )  /* not equal
*/
    {
/*  022499
    strcpy( qmsg, "??? Qname is not equal            ");
    EXEC CICS SEND FROM (qmsg) LENGTH(80) ERASE;

  */

    strncpy(PRE_QNAME, MsgDesc.ReplyToQ, MQ_Q_NAME_LENGTH);
          strncpy(ObjDesc.ObjectName, MsgDesc.ReplyToQ, MQ_Q_NAME_LENGTH);
         strncpy(ObjDesc.ObjectQMgrName, MsgDesc.ReplyToQMgr,
MQ_Q_MGR_NAME_LENGTH);
   ObjDesc.ObjectType = MQOT_Q;
   memcpy(&ObjDesc.StrucId , MQOD_STRUC_ID, sizeof(MQOD_STRUC_ID));
   ObjDesc.Version = MQOD_VERSION_1;


         OpenOptions = MQOO_OUTPUT +
                       MQOO_FAIL_IF_QUIESCING;

         MQOPEN(p_hconn,
                &ObjDesc,
                OpenOptions,
                &p_handle,
                &CompCode,
                &Reason);

         /*****************************************************************/
         /* Test the output from the open, if not OK then build an error   */
         /* message                                                        */
         /*****************************************************************/

/* 022499
   strcpy( qmsg, "??? After open put-code      ");
   EXEC CICS SEND FROM (qmsg) LENGTH(80) ERASE;

         sprintf(qmsg, qmsg,CompCode,Reason);
   EXEC CICS SEND FROM (qmsg) LENGTH(80) ERASE;

         strncpy(qmsg, ObjDesc.ObjectName,  MQ_Q_NAME_LENGTH);
   EXEC CICS SEND FROM (qmsg) LENGTH(80) ERASE;

         strncpy(qmsg, ObjDesc.ObjectQMgrName,  MQ_Q_MGR_NAME_LENGTH);
   EXEC CICS SEND FROM (qmsg) LENGTH(80) ERASE;

  */
         if (CompCode == MQCC_FAILED)
         {
         strncpy(Operation, "MQOPEN output", sizeof(Operation));
         strncpy(ObjName, ObjDesc.ObjectName, MQ_Q_NAME_LENGTH);
         RecordCallError();
         }
    }   /* END of checking ReplyToQ name   */
   /*****************************************************************/
   /* Set up Put message options for Client local queue              */
   /*****************************************************************/

   PutMsgOpts.Options = MQPMO_NO_SYNCPOINT +
               MQPMO_FAIL_IF_QUIESCING;


   /*****************************************************************/
   /* Set up Object descriptor for Client local queue or ReplyToQ   */
   /*****************************************************************/
   /*   PUT the message to local queue    */
   /*
   ObjDesc.ObjectType = MQOT_Q;
   strncpy(ObjDesc.ObjectName, PUTQ_QNAME, MQ_Q_NAME_LENGTH);
   */
   /*   PUT the message to a Reply to Queue  */
   ObjDesc.ObjectType = MQMT_REPLY;
   strncpy(ObjDesc.ObjectName, MsgDesc.ReplyToQ, MQ_Q_NAME_LENGTH);
   strncpy(ObjDesc.ObjectQMgrName, MsgDesc.ReplyToQMgr, MQ_Q_MGR_NAME_LENGTH);



   /****************************************************************/
   /* Set up Message descriptor for Client local queue               */
   /****************************************************************/

   memcpy(MsgDesc.CorrelId, MsgDesc.MsgId, sizeof(MsgDesc.MsgId));

   /****************************************************************/
   /* Put the message to the Client local queue                      */
   /****************************************************************/
   PutBuffLen  = 400;
   memcpy(PutBuffer, &MsgBuffer, MsgBuffLen);

   if (!strncmp(MsgDesc.Format, "MQCICS", 6) )
   {
       memcpy(PutBuffer+164, mqdata, 70);
   }
   else
   {
       memcpy(PutBuffer, mqdata, 70);
   }



/*
  MQPUT1(Hconn,
          &ObjDesc,
          &MsgDesc,
          &PutMsgOpts,
          PutBuffLen,
          &PutBuffer,
          &CompCode,
          &Reason);
*/


   MQPUT(p_hconn,
    p_handle,
          &MsgDesc,
          &PutMsgOpts,
          PutBuffLen,
          &PutBuffer,
          &CompCode,
          &Reason);


      if (CompCode == MQCC_FAILED)
         {
        strncpy(Operation, "MQPUT TO CLIENT/ReplyToQ Q", sizeof(Operation));
        /* strncpy(ObjName, PUTQ_QNAME, MQ_Q_NAME_LENGTH);  */
     strncpy(ObjName, MsgDesc.ReplyToQ, MQ_Q_NAME_LENGTH);
        RecordCallError();
         }
   return;
   }




/*****************************************************************/
/* This function writes two messages to the CICS operator defining   */
/* the problem that has occurred. The failed Operation and Object    */
/* Name fields are completed by the calling function.  The remaining */
/* fields of the message are completed by this routine.              */
/*****************************************************************/

void RecordCallError(void)
   {

   char Abstime[8];
   char Date[8];
   char Time[8];
   char Transaction[4];
   long TaskNum;

   short MsgLen;

   /*************************************************************/
   /* Obtian information about transaction                         */
   /*************************************************************/

   EXEC CICS ASKTIME
        ABSTIME(Abstime);

   EXEC CICS FORMATTIME
        ABSTIME(Abstime)
        DATE(Date) DATESEP
        TIME(Time) TIMESEP;

   strncpy(Transaction, dfheiptr->eibtrnid, sizeof(Transaction));

   /*************************************************************/
   /* Call function to convert the packed decimal number into       */
   /* unpacked number                                              */
   /*************************************************************/

   PackToLong((unsigned char *)dfheiptr->eibtaskn,
              sizeof(dfheiptr->eibtaskn), &TaskNum);

   /*************************************************************/
   /* Format first error message                                   */
   /*  - "An error has occurred in <Tran> <Task#> <Date> <Time>"    */
   /*************************************************************/

   MsgLen = (short) sprintf(ErrorMsg, ERROR_MSG_1, Transaction,
                            TaskNum, Date, Time);
   EXEC CICS WRITEQ TD
        QUEUE("CSMT")
        FROM(ErrorMsg)
        LENGTH(MsgLen);

   /*****************************************************************/
   /* Format second error message                                   */
   /* -"ERROR - Operation <xxx> CompCode <x> Reason <x> Object <x>" */
   /*****************************************************************/

   MsgLen = (short) sprintf(ErrorMsg, ERROR_MSG_2, Operation,
                            CompCode, Reason, ObjName);

   EXEC CICS WRITEQ TD
        QUEUE("CSMT")
        FROM(ErrorMsg)
        LENGTH(MsgLen);

   return;
   }



/*********************************************************************/
/*  This function forwards a message to the sample's dead letter     */
/*  queue.                                                           */
/*********************************************************************/

void ForwardMsgToDLQ(void)
   {
   PMQDLH pDLQHdr;

   /*****************************************************************/
   /* Set up Put message options for dead letter queue              */
   /*****************************************************************/

   PutMsgOpts.Options = MQPMO_NO_SYNCPOINT +
                        MQPMO_PASS_IDENTITY_CONTEXT;
   PutMsgOpts.Context = Hobj_InputQ;

   /*****************************************************************/
   /* Set up dead letter message data                               */
   /*****************************************************************/

   PutBuffLen  = min(DataLen, MsgBuffLen);
   PutBuffLen  = min(PutBuffLen, 2000);
   memcpy( DLQBuffer.MessageData,
           &MsgBuffer,
           (size_t) PutBuffLen);

   /*****************************************************************/
   /* Set up dead letter message header                             */
   /*****************************************************************/

   pDLQHdr = &DLQBuffer.DeadLetterHeader;

   memcpy( pDLQHdr->StrucId , MQDLH_STRUC_ID , sizeof(MQCHAR4) );
   pDLQHdr->Version = MQDLH_VERSION_1;
```

```
if (bBadMsg == FALSE)
{
  pDLQHdr->Reason  = Reason;
  memcpy( pDLQHdr->DestQName,
          ObjDesc.ObjectName,
          MQ_Q_NAME_LENGTH );
}
else
{
  pDLQHdr->Reason  = MQFB_XMIT_Q_MSG_ERROR;
  memset( pDLQHdr->DestQName, ' ', MQ_Q_NAME_LENGTH );
};

memset( pDLQHdr->DestQMgrName, ' ', MQ_Q_NAME_LENGTH );
pDLQHdr->Encoding       = MsgDesc.Encoding;
pDLQHdr->CodedCharSetId = MsgDesc.CodedCharSetId;
memcpy( pDLQHdr->Format , MsgDesc.Format , MQ_FORMAT_LENGTH );
pDLQHdr->PutApplType = MQAT_CICS;
memcpy( pDLQHdr->PutApplName , APPL_NAME,
        MQ_PUT_APPL_NAME_LENGTH );

/*****************************************************************/
/* Set up Object descriptor for dead letter queue                */
/*****************************************************************/

ObjDesc.ObjectType = MQOT_Q;
strncpy(ObjDesc.ObjectName, DEAD_QNAME, MQ_Q_NAME_LENGTH);

/*****************************************************************/
/* Set up Message descriptor for dead letter queue               */
/*****************************************************************/

MsgDesc.Persistence = MQPER_PERSISTENCE_AS_Q_DEF;
memcpy( MsgDesc.Format
      , MQFMT_DEAD_LETTER_HEADER
      , MQ_FORMAT_LENGTH
      );

/*****************************************************************/
/* Put the message to the dead letter queue                      */
/*****************************************************************/

MQPUT1(Hconn,
       &ObjDesc,
       &MsgDesc,
       &PutMsgOpts,
       PutBuffLen,
       &DLQBuffer,
       &CompCode,
       &Reason);

strncpy(Operation, "MQPUT TO DLQ", sizeof(Operation));
strncpy(ObjName, DEAD_QNAME, MQ_Q_NAME_LENGTH);
RecordCallError();

return;
}
```

```
/*******************************************************************/
/*  This function converts the CICS packed decimal task number to    */
/*  a long integer.                                                  */
/*******************************************************************/

void PackToLong(unsigned char *source,
                  unsigned int source_size,
                  long int *target)
   {

   static const unsigned char lmask = 0x0f;

   unsigned int i;
   long int result = 0;
   unsigned int high_nibble, low_nibble;

   for (i = 0; i < source_size-1; i++)
      {
      high_nibble = ((unsigned int)(source[i]>>4))&lmask;
      result *= 10;
      result += (long) high_nibble;

      low_nibble = (unsigned int) source[i]&lmask;
      result *= 10;
      result += (long) low_nibble;
      }

   high_nibble = ((unsigned int)(source[i]>>4))&lmask;
   result *= 10;
   result += (long) high_nibble;

   low_nibble = (unsigned int) (source[i]&lmask);
   if (low_nibble == 0xd)
      {
      result *= -1;
      }

   *target = result;

   return;
   }
```

Appendix D - CPIT CICS Source Code Examples

The makefile:

```
#
# NAME:         esql_msxa.mak
# VERSION:      2.0
# COPYRIGHT:
#
#   (C) COPYRIGHT International Business Machines Corp. 1993, 1998
#   All Rights Reserved
#   Licensed Materials - Property of IBM
#
#   US Government Users Restricted Rights - Use, duplication or
#   disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
#
#            NOTICE TO USERS OF THE SOURCE CODE EXAMPLES
#
# INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THE SOURCE CODE
# EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, "AS IS"
# WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING,
# BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
# FITNESS FOR A PARTICULAR PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY
# AND PERFORMANCE OF THE SOURCE CODE EXAMPLES, BOTH INDIVIDUALLY AND
# AS ONE OR MORE GROUPS, IS WITH YOU.  SHOULD ANY PART OF THE SOURCE
# CODE EXAMPLES PROVE DEFECTIVE, YOU (AND NOT IBM) ASSUME THE ENTIRE
# COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
#
#
#
#
#  This makefile allows the construction of an MS SQL Server Switchloadfile
#  for an embedded SQL oriented environment.
#
#  NOTE : You should change CICSPATH, MSDTCPATH, DBLIBPATH and ESQLPATH to
#       : reflect your installation
#
# This makefile should be run with the command "nmake -f esql_msxa.mk"
#
#
#  Note:
#  The linker will return three
#  "LINK4049 warnings: locally defined symbol <symbol> imported" for
#  the symbols _CICSMSCONNECT, _CICSMSGETCONNECT, and _msqlsrvxa1.
#  This is OK.


# Compiler and linker

CC=cl
LINK=link
MSDTCPATH=d:\mssql\msdtc
ESQLPATH=d:\mssql\esql
DBLIBPATH=d:\mssql\dblib
CICSPATH=e:\opt\cics
ENCINC=e:\opt\encina\include
ENCLIB=e:\opt\encina\lib
```

```
CFLAGS=-Z7 -Od -c -W3 -MD -Gz -DWIN32 -I$(ENCINC)
LINKFLAGS=-dll -nodefaultlib:libc.lib
LINKFLAGS1=-dll
LIBS= $(CICSPATH)\lib\libcicsrt.lib kernel32.lib user32.lib wsock32.lib
advapi32.lib $(ENCLIB)\libEncServer.lib $(ENCLIB)\libEncina.lib netapi32.lib
$(DBLIBPATH)\lib\ntwdblib.lib $(ESQLPATH)\lib\sqlakw32.lib
$(ESQLPATH)\lib\caw32.lib

all :  esql_msxa.dll

esql_msxa.obj : esql_msxa.sqc;
   set include=$(ESQLPATH)\include;$(DBLIBPATH)\include;$(INCLUDE)
   nsqlprep esql_msxa.sqc /NOSQLACCESS
   $(CC) $(CFLAGS) esql_msxa.c

regxa_dblib.dll : $(CICSPATH)\lib\regxa_dblib.obj esql_msxa.obj;
   $(LINK) $(LINKFLAGS1) $(CICSPATH)\lib\regxa_dblib.obj esql_msxa.obj
-out:regxa_dblib.dll $(LIBS) -IMPLIB:regxa_dblib.lib
   xcopy regxa_dblib.dll $(CICSPATH)\bin
   xcopy regxa_dblib.lib $(CICSPATH)\lib

esql_msxa.dll : regxa_dblib.dll $(CICSPATH)\lib\regxa_msswxa.obj;
   $(LINK) $(LINKFLAGS) $(CICSPATH)\lib\regxa_msswxa.obj -out:esql_msxa.dll
$(LIBS) regxa_dblib.lib $(MSDTCPATH)\lib\xaswitch.obj -IMPLIB:esql_msxa.lib
```

The DLL source:

```
/*
* NAME:         esql_msxa.sqc
* VERSION:      2.0
* COPYRIGHT:
*
*   (C) COPYRIGHT International Business Machines Corp. 1993, 1997
*   All Rights Reserved
*   Licensed Materials - Property of IBM
*
*   US Government Users Restricted Rights - Use, duplication or
*   disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
*
*
*             NOTICE TO USERS OF THE SOURCE CODE EXAMPLES
*
* INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THE SOURCE CODE
* EXAMPLES, BOTH INDIVIDUALLY AND AS ONE OR MORE GROUPS, "AS IS"
* WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING,
* BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
* FITNESS FOR A PARTICULAR PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY
* AND PERFORMANCE OF THE SOURCE CODE EXAMPLES, BOTH INDIVIDUALLY AND
* AS ONE OR MORE GROUPS, IS WITH YOU.  SHOULD ANY PART OF THE SOURCE
* CODE EXAMPLES PROVE DEFECTIVE, YOU (AND NOT IBM) ASSUME THE ENTIRE
* COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
*
*
*
```

```
* This is the source for building MS SQL 6.5 XA support into CICS where
* the environment is embedded SQL oriented.
*
*
*/

/* define the DBPROCESS pointer */
#define DBPROCESS void
#define PDBPROCESS DBPROCESS*

#include <stdio.h>          /* standard C runtime header */
#include <stdlib.h>         /* standard C runtime header */

/* internal variables */
FILE *fTrace;        /* trace file */
char szBuf[256];     /* generic char buffer */


/* declare embedded SQL host variables */
EXEC SQL BEGIN DECLARE SECTION;

char    szLogin[30];
char    szSQLServerDB[30];
char    szConnectionName[30];

DBPROCESS *pDbpCICS = NULL;
long SQLCODE;

EXEC SQL END DECLARE SECTION;

/*****************************************************************************
*
//
//  FUNCTION: CICSMSCONNECT()
//
//      Establish a long-running, NAMED connection to Microsoft SQL Server.
//      Programs will use this established connection when they interact with
//      SQL Server. The XAD Product Definition's resource manager
//       initialization string contains the values that are passed into this
//       function as parameters when XA_OPEN is called.
//
//  PARAMETERS:
//
//      nConnectionId - a unique integer to identify a DBPROCESS handle
//
//      pszConnectionName - arbitrary NAMED connection name. CICS programs
//         should embed a SET CONNECTION TO <NAME> to reference the connection

//
//      pszLogin - SQL Server login and password in the form <user>.<password>
//
//      pszServerDB - SQL Server configured client name and default database
//        in the form <configured client name>.<database>
//
//  RETURNS:
//
//      0 if successful, non-zero SQLCODE if error occurs
//
//  COMMENTS:
```

```
//       CICS will close all connections automatically when the region is
//       stopped.
//
//       Please see CICS Administration Guide, Microsoft Emebedded SQL for C
//       documentation, and SQL Server Books Online for more details.
******************************************************************************
/
__declspec(dllexport) int __cdecl CICSMSCONNECT(
     int nConnectionId,                  /* connection id */
     char *pszConnectionName,            /* NAMED connection name */
     int clen,                                 /* strlen of previous
parameter */
     char *pszLogin,                     /* login name and password */
     int ulen,                                 /* strlen of previous
parameter */
     char *pszSQLServerDB,               /* server and default database */
     int slen)                                 /* strlen of previous
parameter */
{
   /* name of trace file is in the form cicsmssql.<connectionid> */
   sprintf(szBuf,"cicsmssql.%d", nConnectionId);

    /* attempt to open trace file */
   if ((fTrace = fopen(szBuf, "a")) == ((void *)0))
    {
       return(5);    /* fopen failed, bail out */
    }

    /* mark beginning of current trace */
   fprintf(fTrace, "CICSMSCONNECT, attempting connection to SQL Server...\n");
   fflush(fTrace);   /* make sure it gets written */

    /* write entry into trace file */
   fprintf(fTrace,
              "ConnectId: %d, ConnectName: %s, Login: %s, Server.DB: %s\n",
           nConnectionId,pszConnectionName,pszLogin,pszSQLServerDB);
   fflush(fTrace);   /* make sure it gets written */

    /* copy passed in parameters to global host variables */
   strcpy(szLogin, pszLogin);
    strcpy(szSQLServerDB, pszSQLServerDB);
    strcpy(szConnectionName, pszConnectionName);

    /* embedded sql to make database connection */
    EXEC SQL CONNECT TO :szSQLServerDB
    AS :szConnectionName USER :szLogin;

    /* check for successful connection */
   if(SQLCODE == 0)
    {
       /* write entry into trace file */
         fprintf(fTrace, "Connection established!\n");
         fclose(fTrace);   /* close the trace file */

         return(0);  /* return a zero */
    }
   else  /* connection failed */
   {
       /* write entry into trace file */
```

```
        fprintf(fTrace,
                    "Connection attempt failed. SQLCODE: %d\n",
                    SQLCODE);
        fclose(fTrace);   /* close the trace file */

        return(SQLCODE);  /* return non-zero value */
   }
}

/******************************************************************************
*
//
//  FUNCTION: CICSMSGETCONNECT()
//
//      Reference an established long-running, NAMED connection to Microsoft
//       SQL Server. The XAD Product Definition's resource manager
//       initialization string contains the values that are passed into this
//       function as parameters when XA_OPEN is called.
//
//  PARAMETERS:
//
//      nConnectionId - a unique integer to identify a DBPROCESS handle
//
//      pszConnectionName - arbitrary NAMED connection name. CICS programs
//         should embed a SET CONNECTION TO <NAME> to reference the connection

//
//      pDbProcess - a pointer to the address of a DBPROCESS structure.The
//             DBPROCESS structure is the basic data structure that DB-Library

//             uses to communicate with SQL Server
//
//  RETURNS:
//
//      0 if successful, non-zero SQLCODE if error occurs
//
//  COMMENTS:
//
//       Please see CICS Administration Guide, Microsoft Emebedded SQL for C
//       documentation, and SQL Server Books Online for more details.
******************************************************************************
/
__declspec(dllexport) int __cdecl CICSMSGETCONNECT(
   int nConnectionId,                    /* connection id */
   char *pszConnectionName,            /* NAMED connection name */
    int clen,                            /* strlen of previous parameter */
    PDBPROCESS *pDbProcess)              /* pointer to a DBPROCESS pointer */
{

   /* name of trace file is in the form cicsmssql.<connectionid> */
   sprintf(szBuf,"cicsmssql.%d", nConnectionId);

    /* attempt to open trace file */
   if ((fTrace = fopen(szBuf, "a")) == ((void *)0))
    {
       return(5);    /* fopen failed, bail out */
    }

    /* mark beginning of current trace */
   fprintf(fTrace, "CICSMSGETCONNECT, Attempting to get connection
pointer...\n");
   fflush(fTrace);   /* make sure it gets written */

    /* copy passed in parameter to global host variable */
    sprintf(szConnectionName, "%s", pszConnectionName);

   /* embedded SQL to get DBPROCESS pointer for specified connection */
    EXEC SQL GET CONNECTION :szConnectionName INTO :pDbpCICS;

   /* check for successful retrieval of pointer */
    if(pDbpCICS != NULL && SQLCODE == 0)
    {
         *pDbProcess = pDbpCICS; /* copy pointer value to caller's address */

        /* write entry into trace file */
          fprintf(fTrace,"Got DBPROCESS connection, value is
%p.\n",*pDbProcess);
          fclose(fTrace);   /* close the trace file */

          return(0);  /* return a zero */
    }
   else  /* unsuccessful */
   {
       /* write entry into trace file */
         fprintf(fTrace,
               "Failed to retrieve DBPROCESS connection. SQLCODE: %d\n",
               SQLCODE);
         fclose(fTrace);   /* close the trace file */

      return(SQLCODE);           /* return the SQLCODE */
   }
}
```

Appendix E - References

IBM DB2 UDB for UNIX Quick Beginnings (S10J-8148)
IBM TXSeries for AIX Quick Beginnings (GC33-1744)
Lotus Notes Release 4.5: A Developer's Handbook (SG24-4876)
Lotus Solutions for the Enterprise, Volume 4 Lotus Notes and the MQSeries Enterprise Integrator (SG24-2217)
MQSeries Application Programming Guide (SC33-0807)
MQSeries Application Programming Reference (SC33-1673)
MQSeries Application Programming Reference Summary (SC33-6095)
MQSeries Command Reference (GC33-1369)
MQSeries for AIX V5.0 Quick Beginnings (GC33-1867)
MQSeries for OS/2 Warp Quick Beginnings (GC33-1868)
MQSeries for Sun Solaris Quick Beginnings (GC33-1870)
MQSeries for Windows NT Quick Beginnings (GC33-1871)
MQSeries Intercommunications (SC33-1872)
MQSeries Messages (GC33-1876)
MQSeries System Administration (SC33-1873)

IBM CICS Product Homepage: http://www.software.ibm.com/ts/cics
IBM DB2 Product Family Homepage: http://www.software.ibm.com/data/db2/
IBM eNetwork Communications Server Product Homepage: http://www.software.com/network/commserver
IBM MQSeries Product Homepage: http://www.software.ibm.com/ts/mqseries/
IBM MQSeries SupportPacs: http://www.software.ibm.com/ts/mqseries/txppacs
IBM Software Homepage: http://www.software.ibm.com
IBM Software Product Index: http://www2.software.ibm.com/prodindex/prodindex.nsf/webalphaview?OperView
IBM TXSeries Product Homepage: http://www.software.ibm.com/ts/txseries
Lotus Homepage: http://www.lotus.com
Microsoft Homepage: http://www.microsoft.com
Netscape Homepage: http://www.netscape.com

Appendix F - Disclaimer and Trademarks

Disclaimer

The information in this document is distributed as is. The use of this information or the implementation of this configuration and the associated techniques is an individual customer responsibility, and depends upon each customer's ability to evaluate the information and techniques and incorporate them into its own individual business situation. Customers attempting to adapt these techniques to the own environment do so at their own risk. IBM accepts no liability for consequences resulting from the use of this information. This report is to be used solely for installing and supporting the products and configuration discussed. This material must not be used for other purposes without the written consent of IBM.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries. Those trademarks followed by an asterisk(*) are registered trademarks of the International Business Machines Corporation.

Trademarks

AIX*
AS/400*
CICS*
DB2*
DB2* Client Application Enablers
DB2* Connect
DCE
DRDA*
eNetwork
IBM*
IMS*
MQEI
MQLSX
MQSeries*
OS/390
OS/2
OS/2 WARP Server
RS/6000
S/390*
SupportPac
TXSeries
VisualAge*

The following terms are trademarks of the Lotus Corporation:
Domino
Lotus
Lotus Notes