28

MQ

October 2001

In this issue

© Xephon plc 2001

update

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38126
From USA: 01144 1635 38126
Fax: 01635 38345
E-mail: info@xephon.com

North American office

Xephon/QNA
Post Office Box 350100
Westminster CO 80035-0100, USA
Telephone: (303) 410 9344
Fax: (303) 438 0290

Contributions

When Xephon is given copyright, articles published in *MQ Update* are paid for at the rate of £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a flat fee of £30 ($50) per article. For more information about contributing an article you can download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

MQ Update on-line

Code from *MQ Update*, and complete issues in Acrobat PDF format, can be downloaded from our Web site at www.xephon.com/mq; you will need to supply a word from the printed issue.

Commissioning Editor

Peter Toogood
E-mail: PeterT@xephon.net

Managing Editor

Madeleine Hudson
E-mail: MadeleineH@xephon.com

Disclaimer

Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, scripts, and other contents of this journal before making any use of it.

Subscriptions and back-issues

A year's subscription to *MQ Update*, comprising twelve monthly issues, costs £255.00 in the UK; $380.00 in the USA and Canada; £261.00 in Europe; £267.00 in Australasia and Japan; and £265.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the July 1999 issue, are available separately to subscribers for £22.50 ($33.75) each including postage.

© Xephon plc 2001. All rights reserved. None of the text in this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior permission of the copyright owner. Subscribers are free to copy any code reproduced in this publication for use in their own installations, but may not sell such code or incorporate it in any commercial product. No part of this publication may be used for any form of advertising, sales promotion, or publicity without the written permission of the publisher. Copying permits are available from Xephon in the form of pressure-sensitive labels, for application to individual copies. A pack of 240 labels costs $36 (£24), giving a cost per copy of 15 cents (10 pence). To order, contact Xephon at any of the addresses above.

Printed in England.

MQSI V2 Service Trace

MQSI V2 has some excellent trace facilities. This article explains how to take an MQSI V2 Service Trace for the Configuration Manager. It should then be possible to see how to run an MQSI V2 Service Trace for other components.

The MQSI V2 User Trace facility is excellent for diagnosing problems with message flows and, in some cases, databases.

The MQSI V2 Service Trace facility is excellent for diagnosing 'system'-related problems concerning MQSI V2 components.

A problem often encountered is the inability to deploy to an MQSI V2 Broker because of an inconsistency in the MQSI V2 Configuration Manager, resulting in the following message:

```
BIP1503E: Unable to find required message set document in the
configuration repository. When deploying configuration data, the
document for message set '849e4b21-e400-0000-0080-8ca40b81b5fa'could not
be found in the configuration repository. Another document has
referenced this document, and it is required for the successful
completion of the deploy operation. The referencing document is
broker'BRKNTD0A'.
The message set document '849e4b21-e400-0000-0080-8ca40b81b5fa' has
either been deleted or has not been checked in. Correct the problem and
retry the deploy operation.
08:55 19/01/2001
```

It was simple to detect which MessageSet was causing the problem by using the Service Trace. The clue was the identifier *849e4b21-e400-0000-0080-8ca40b81b5fa*.

After running the MQSI V2 Service Trace against the Configuration Manager, looking at the output, and searching for 'error', it was obvious which MessageSet was preventing the deploy operation from being successful.

Here is how the Service Trace was run from the command prompt:

```
mqsichangetrace ConfigMgr -t -b -l debug
Execute Deploy Operation
mqsireadlog ConfigMgr -t -b agent -f -o servicetrace.tm$
mqsiformatlog  -i servicetrace.tm$ -o servicetrace.txt
mqsichangetrace ConfigMgr -t -b -l none
```

© 2001. Reproduction prohibited. Please inform Xephon of any infringement.

Commands one, three, four, and five might be put into a script in order to make the Service Trace easier to run. A pause is required at command two, in order to execute the deploy to the MQSI V2 Broker.

Apart from resolving problems, it is possible to learn much about how the Configuration Manager or Broker work together with MQSeries.

Here is a sample of some of the trace entries you might expect to see.

```
{CICON=images/MessageProcessingNodeType.gif, CLASTUPDATE=2000-09-21
13:45:00.443002, CUUID=5c0d03be-e100-0000-0080-9ed6221cc8e3,
CSECTION=SHARED, CXMLDATA=<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE MessageProcessingNodeType SYSTEM "mqsi.dtd" >
<MessageProcessingNodeType scaleableIcon="" longDescription=""
icon="images/MessageProcessingNodeType.gif" versionCreator="" package=""
shortDescription="" version="" collectionPath="" creationTimestamp=""
isPrimitive="false" creator="" versionTimestamp="" xmi.uuid="5c0d03be-
e100-0000-0080-9ed6221cc8e3" xmi.id="5c0d03be-e100-0000-0080-
9ed6221cc8e3" xmi.label="BOIFLOW">
  <Connection source="42495dbe-e100-0000-0080-9ed6221cc8e3.ErrorFixed"
longDescription="" icon="images/Connection.gif" versionCreator=""
shortDescription="" creationTimestamp="" creator="" target="8fa505be-
e100-0000-0080-9ed6221cc8e3.in" versionTimestamp=""
xmi.label="Connection9">
2001-03-22 07:43:08.090       424}
com.ibm.broker.config.BrokerManager@952c75.run ConfigMgr ConfigMgr ,
ConfigMgr
2001-03-22 07:43:17.493       419    ImbQueue::read MQSeries returnCode
and reasonCode , 2, 2161
2001-03-22 07:43:17.493       419    ImbQueue::read Unable to mqget from
Queue , SYSTEM.BROKER.SECURITY.REPLY, CLIFTON
2001-03-22 07:43:17.503       419    ImbQueue::read file:f:/build/argo/
src/AdminAgent/ImbQueue.cpp line:463 message:2070.MQSeriesIntegrator2
Unable to get from Queue , MQGET, SYSTEM.BROKER.SECURITY.REPLY, CLIFTON,
2, 2161
2001-03-22 07:43:17.503       419      Error       BIP2070E: A problem
was detected with MQSeries while issuing MQGET for MQSeries queue
SYSTEM.BROKER.SECURITY.REPLY, MQSeries queue manager CLIFTON.  MQCC=2,
MQRC=2161.
```

The operation on the specified queue or queue manager returned with the indicated MQSeries completion and reason code.

Check the MQSeries completion and reason codes in the *MQSeries Application Programming Reference Manual* to establish the cause of the error, taking any appropriate action. It may be necessary to restart the message broker after you have performed this recovery action. If an MQOPEN was unsuccessful because the queue manager or queue

© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

did not exist, then define these objects to MQSeries. If the problem was because incorrect object names were specified, then the message broker will try to recover. If the problem persists, it may be necessary to restart the message broker.

```
2001-03-22 07:43:17.503       419      {
ImbQueue::generalErrorProcessing , 2, 2161
2001-03-22 07:43:17.503       419        {
ImbQueueManager::endUnitOfWork , false
2001-03-22 07:43:17.503       419        ImbQueueManager::endUnitOfWork
MQSeries returnCode and reasonCode , 0, 0.
```

Ken Marshall
MQSeries Consultant, MQSolutions (UK) © MQSolutions

Adding user data in MQSeries send channel exits

WHAT ARE SEND CHANNEL EXITS?

If the user correctly defines a send channel exit, all data that flows across the channel is passed to it for modification immediately before it is passed to the communications protocol for transmission. The send exit may, for instance, apply a compression algorithm to the data and return the compressed data to the MQSeries channel code, which then transmits it. In the case of a compression algorithm, this results in fewer bytes being sent across the communication protocol. If the send exit modifies the data, it is, of course, essential that the modification is reversed at the receiving end of the channel. This is achieved by specifying an appropriate receive exit at that end to reverse the algorithm applied by the send exit.

SEND EXITS COULDN'T MAKE THE DATA LARGER

The compression example described above leaves the data smaller than it was before the exit was called and could have been implemented on any version of the MQSeries messaging product which supports send exits. But, prior to version MQSeries V5.2, it was not possible to implement an algorithm which increased the size of the data. In

© 2001. Reproduction prohibited. Please inform Xephon of any infringement.

particular, this restriction arose because MQSeries messages are often longer than the maximum protocol transmission size (which is generally 32K for TCP/IP). They are, therefore, divided into segments to be sent across the transmission medium (note that this protocol-level segmentation is transparent to the MQSeries application programmer and is quite distinct from the message segmentation which can be invoked at the MQSeries application programming interface).

As the send exits are called directly before transmission they are passed individual segments of a segmented message. These segments were produced such that they completely filled the maximum transmission buffer size, so there was no room for the user to add data.

WHY WRITE A SEND EXIT WHICH MAKES THE DATA LARGER?

So, prior to V5.2, customers were not able to add to the data in a send exit. But why would a user want to add extra information to the data anyway? In general, the demand to add data relates to security applications – perhaps a key is to be flowed with the data, or the user wants to add a digital signature.

Digital signatures ensure data integrity and authentication of the sender for users who are working in an environment where public/ private key pairs are being used. This is commonly referred to as Public Key Infrastructure (or PKI) security. The text is hashed by the sender according to an algorithm known by the receiver. The resulting hash value is then encrypted by the sender using the sender's private key. The result is the digital signature for the text. This is appended at the end of the plain text before it is sent.

The receiver has the public key, which matches the sender's private key. The receiver uses this to decrypt the digital signature received and, hence, to recreate the sender's hash value. The hashing algorithm that the receiver applies to the received plain text is the same as the sender's algorithm, so this generates another copy of the sender's hash value. If the two hash values match, the receiver knows that the plain text in the message has not been corrupted or tampered with in transit, and that the sender is as expected.

© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

So how does the user add data?

With MQSeries V5.2 a new parameter called ExitSpace is allowed in the MQCXP structure which is passed to MQSeries channel exit programs. ExitSpace is an output parameter that the user code in the send exit sets on exit initialization: the parameter is not used on other types of channel exit.

An example of the code required to set the ExitSpace field is as follows:

```
void MQENTRY EXAMPLE_SENDEXIT (
        PMQVOID  pChannelExitParms,   /* Channel exit parameter block */
        PMQVOID  pChannelDefinition,  /* Channel definition           */
        PMQLONG  pDataLength,         /* Length of data               */
        PMQLONG  pAgentBufferLength,  /* Length of agent buffer       */
        PMQVOID  pAgentBuffer,        /* Agent buffer                 */
        PMQLONG  pExitBufferLength,   /* Length of exit buffer        */
        PMQPTR   pExitBufferAddr)     /* Address of exit buffer       */
{
    PMQCXP        pParms;
    pParms   = (PMQCXP)pChannelExitParms;
    if (pParms->ExitId==MQXT_CHANNEL_SEND_EXIT)
   {
      if  (pParms->ExitReason == MQXR_INIT)
      {
         pParms->ExitResponse = MQXCC_OK;
         pParms -> ExitSpace = 16;
         return;
      }
   }
}
```

When the send exit is subsequently invoked to send data this parameter is used to determine the amount of space that will be left free for user data. Of course, the matching receive exit must be written to expect the data, to use it, and to remove it. Examples of send exits adding text, and receive exits removing it, are included below.

```
void MQENTRY EXAMPLE_SENDEXIT (
        PMQVOID  pChannelExitParms,   /* Channel exit parameter block */
        PMQVOID  pChannelDefinition,  /* Channel definition           */
        PMQLONG  pDataLength,         /* Length of data               */
        PMQLONG  pAgentBufferLength,  /* Length of agent buffer       */
        PMQVOID  pAgentBuffer,        /* Agent buffer                 */
        PMQLONG  pExitBufferLength,   /* Length of exit buffer        */
        PMQPTR   pExitBufferAddr)     /* Address of exit buffer       */
{
```

© 2001. Reproduction prohibited. Please inform Xephon of any infringement.

```
    PMQCXP        pParms;
    pParms   = (PMQCXP)pChannelExitParms;
.....................................
.....................................
   if (pParms->ExitId == MQXT_CHANNEL_SEND_EXIT)
   {
 ..................
.....................
      if  (pParms->ExitReason == MQXR_XMIT)
      /* send exit with data */
      msg += *pDataLength;   /* jump over message data */
      for (i = Ø; i <16; i++)
        *(msg + i) = '!';
     *pDataLength += 16;
      pParms->ExitResponse = MQXCC_OK;
      pParms->ExitResponse2 = MQXR2_USE_AGENT_BUFFER;
      return;
   }
}
void MQENTRY EXAMPLE_RECEIVEEXIT (
        PMQVOID  pChannelExitParms,   /* Channel exit parameter block */
        PMQVOID  pChannelDefinition,  /* Channel definition           */
        PMQLONG  pDataLength,         /* Length of data               */
        PMQLONG  pAgentBufferLength,  /* Length of agent buffer       */
        PMQVOID  pAgentBuffer,        /* Agent buffer                 */
        PMQLONG  pExitBufferLength,   /* Length of exit buffer        */
        PMQPTR   pExitBufferAddr)     /* Address of exit buffer       */
{
    PMQCXP        pParms;
    pParms   = (PMQCXP)pChannelExitParms;
   if (pParms->ExitId==MQXT_CHANNEL_RCV_EXIT)
   {
     if  (pParms->ExitReason == MQXR_XMIT)
     /* receive exit with data */
...........
...........
 msg += *pDataLength - 1; /* jump to last character of send exit data */
     for (i = Ø; i < 16; i++)
       if (*(msg - i) != '!')
       {
         pParms->ExitResponse = MQXCC_CLOSE_CHANNEL;
         return;
       };
     *pDataLength -= 16;
     pParms->ExitResponse = MQXCC_OK;
     pParms->ExitResponse2 = MQXR2_USE_AGENT_BUFFER;
     return;
   }
 }
```


© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

SOME DETAILED CONSIDERATIONS

Users are allowed to specify multiple send exits, which are called one after the other. In general, these are matched by an equivalent chain of multiple receive exits. Each send exit in the chain can request its own user data area and MQSeries will ensure that the data passed into the send exit chain has adequate room for all the exits.

When working with data prior to transmission the user is not restricted to the amount of user data requested at exit intialization. In fact, all the spare data in a transmission buffer is made available for user data, so the user may often be able to use significantly more data than he initially requested.

Reserving unnecessarily large amounts of user data is not advisable as this will cause large messages to be divided into more segments than necessary and will, therefore, slow transmission. A minimum of 1K of message data must be included in each transmission.

CONCLUSIONS

The ability for users to add data in send exits from MQSeries V5.2 gives users important new options for implementing security using MQSeries channel exits.

Mike Horan
Software Engineer, IBM Hursley (UK)
© IBM UK

E-mail alerts

If you'd like to be notified when new issues of *MQ Update* have been placed on our Web site, you can sign up for our e-mail alert, which provides this service. To sign up, go to http://www.xephon.com/lists.

© 2001. Reproduction prohibited. Please inform Xephon of any infringement.

MQSI exception processing part 2: coding

INTRODUCTION

In the last issue I introduced coding for MQSI exception processing. We should now be ready to code a more sophisticated exception processing flow for MQSI.

To recap briefly – exception handling paths start at:

- A failure terminal (most message processing nodes have these).
- The catch terminal of an MQInput node or the catch terminal of a TryCatch node.

Normally, when an exception happens in a unit of work within the message flow, it is generally sufficient to roll back the message to the input node and start the exception processing there, in the catch node.

In special circumstances, we might need to add a TryCatch node in order to capture some intermediate information within that message flow.

GENERIC EXCEPTION HANDLING FLOW FOR RE-USE

One of the simplest ways to get code re-use in MQSI is to make the message flow a sub-flow, so that it can be incorporated into other message flows within the broker.

We can make our exception handling a sub-flow by duplicating and then removing the main flow, keeping only the exception flow that was connected to the catch terminal of the MQInput node, replacing the MQInput node with an MQInput terminal. We can call it *ExceptionHandlingSubFlow*, as Figure 1 illustrates.

Once we have created the sub-flow and checked it into the repository it can be added to the workspace of any MQSI developer that is connected to the same configuration manager. The developers can then drag the sub-flow into their message flow and connect it to the catch terminal of the MQInput node, as Figure 2 shows.

© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

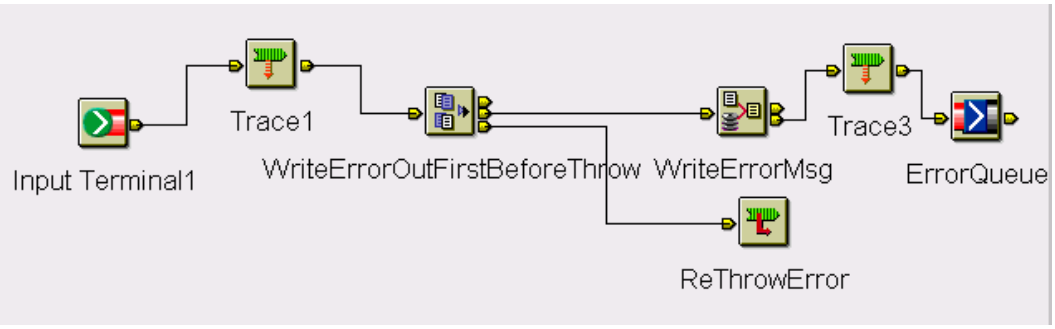

Figure 1: ExceptionHandlingSubFlow

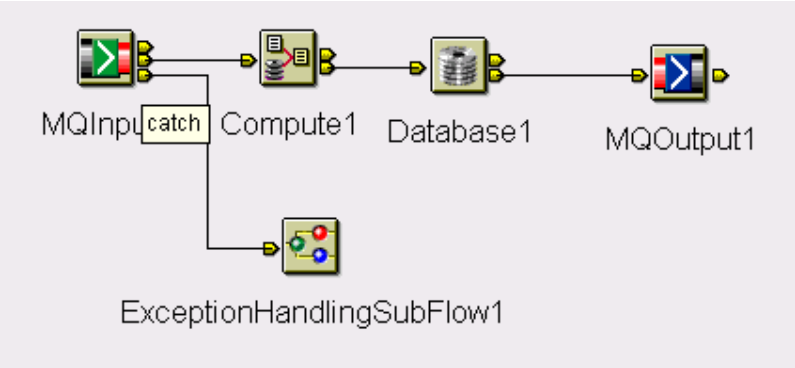

Figure 2: Re-using the sub-flow

In this way, you have standard exception processing for your MQSI developers to use when coding their message flows.

THE STANDARD MQSI EXCEPTION PROCESSING

Now our standard for MQSI exception processing begins to take shape. Procedures within the standard include:

1 Creating a BackOut queue for every input queue of the message flow so that problematic messages can be rolled back and stored for further investigation.

2 Creating an Error queue for the system to hold the error messages generated by the exception flow. Error messages can be related to problematic messages in the Backout queue since they will have the same MessageID.

© 2001. Reproduction prohibited. Please inform Xephon of any infringement.

3 Connecting the *ExceptionHandlingSubFlow* to the catch terminal of the MQInput node.

If your site has some monitoring tools in place you may want to add another step.

4 Monitoring the *curdepth* of the Backout queue and Error queue.

Further enhancement

Now that we have a standard for our MQSI exception processing you can adapt this to your site, or you can further enhance the exception sub-flow.

The MQSI manual describes five types of exception:

- *RecoverableException.*
- *ParserException*.
- *ConversionException.*
- *DatabaseException.*
- *UserException.*

We could further refine our exception processing according to the type of exception that we encounter.

One suggestion is that, if the exception encountered is due to a system error, we should find a way to stop the message flow. Messages are thrown to the BackOut queue with a system error, so we should fix the error before we restart the message flow. In this way, we don't have to re-route all the messages from the BackOut queue into the input queue because the messages were sitting on the input queue when the message flow was stopped. The message flow will simply pick up work once again from where it left off, once the system error is fixed.

An example of such a system error is when the error has an SQL code of *-922*, indicating an authorization error. We should fix the DB2 authorization error before re-starting the message flow.

There are a couple of ways to stop a message flow automatically, without using a monitoring tool.

One way is to mimic an administration message issued by the MQSI

© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

configuration manager that stops the message flow. If you 'right-click' and choose Stop on a message flow from the Operations tab of the MQSI configuration manager, the configuration manager actually puts a stop message into the administration queue *SYSTEM.BROKER.ADMIN.QUEUE* (see Figure 3). All administrative messages are XML messages with the following properties:

- Format of "xml".
- MessageType of *MQMT_REQUEST*.
- *ReplyToQ* – specify your reply to queue.
- *ReplyToQMgr* – specify your queue manager.
- MessageId of *MQMI_NONE*.
- Report of *MQRO_COPY_MSG_ID_TO_CORREL_ID*.
- CharacterSet of 1208 (UTF-8).
- Persistence of *MQPER_PERSISTENCE_AS_Q_DEF*.

The 'stop message flow' message will take the following form:

```
<Broker uuid="1234" label="Broker1" version="1"
     <ExecutionGroup uuid="5678">
          <Stop>
               <MessageFlow uuid="9101112"/>
          </Stop>
     </ExecutionGroup>
</Broker>
```

In order to mimic an administration stop message we need to know the

Figure 3: Stopping the message flow

© 2001. Reproduction prohibited. Please inform Xephon of any infringement.

uu-id for the broker, the execution group, and the message flow before we can issue the stop message to the broker administration queue. This requires quite a number of inquiry messages, reading the response messages, and pasting the information before we can correctly configure the stop message.

Another point to remember when mimicking administration messages is that MQSI is sensitive to version numbers, and messages may differ from one release to another.

An alternative to stopping the message flow automatically from the message flow itself is to issue an MQ PCF command to alter the input queue of the message flow and make it *GetInhibited*. In this way, even though the message flow is still running, it cannot get messages from the input queue specified in the MQInput node so the message flow is as good as stopped.

PCF (PROGRAMMABLE COMMAND FORMATS)

PCFs are described in Part Two of the *MQSeries Programmable System Management* book.

PCFs define command and reply messages that can be exchanged between a program and any queue manager (that supports PCF) in a network. You can use PCF commands in a systems management application program for the administration of MQSeries objects, ie queue managers, process definitions, queues, and channels. The application can operate from a single point in the network to communicate command and reply information with any queue manager, local or remote, via the local queue manager.

Each queue manager has an administration queue with a standard queue name, and your application can send PCF command messages to that queue. Each queue manager also has a command server to service the command messages from the administration queue. PCF command messages can, therefore, be processed by any queue manager in the network and the reply data can be returned to your application using your specified reply queue. PCF command and reply messages are sent and received using the normal Message Queue Interface (MQI).

© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

Each command and its parameters are sent as a separate command message containing a PCF header followed by a number of parameter structures. The PCF header identifies the command and the number of parameter structures that follow in the same message. Each parameter structure provides a parameter to the command.

Replies to the commands generated by the command server have a similar structure.

The change queue PCF command

As described in Chapter eight of the manual, PCF commands can be grouped into the following categories:

- Queue Manager.
- Namelist.
- Process.
- Queue.
- Channel.
- Statistics.
- Escape.
- Cluster.

Within the queue commands, the following actions can be performed:

- Change queue.
- Clear queue.
- Copy queue.
- Create queue.
- Delete queue.

The detail of the change queue command is as follows:

The change queue command – **MQCMD_CHANGE_Q** – changes the specified attributes of an existing MQSeries queue.

- Required parameters

© 2001. Reproduction prohibited. Please inform Xephon of any infringement.

  - *QName, QType.*
- Optional parameters (any QType)
  - *QDesc, InhibitPut, DefPriority, DefPersistence.*
- Optional parameters (alias QType)
  - *Force, InhibitGet, BaseQName, Scope, ClusterName, ClusterNamelist, DefBind.*
- Optional parameters (local QType)
  - *Force, InhibitGet, ProcessName, MaxQDepth, MaxMsgLength, BackoutThreshold.*
  - *BackoutRequeueName, Shareability, DefInputOpenOption, HardenGetBackout.*
  - *MsgDeliverySequence, RetentionInterval, DistLists, Usage.*
  - *InitiationQName, TriggerControl, TriggerType, TriggerMsgPriority.*
  - *TriggerDepth, TriggerData, Scope, QDepthHighLimit, QDepthLowLimit.*
  - *QDepthMaxEvent, QDepthHighEvent, QDepthLowEvent, QServiceInterval.*
  - *QServiceIntervalEvent, ClusterName, ClusterNamelist, DefBind.*
- Optional parameters (remote QType)
  - *Force, RemoteQName, RemoteQMgrName, XmitQName, Scope, ClusterName.*
  - *ClusterNamelist, DefBind.*
- Optional parameters (model QType)
  - *InhibitGet, ProcessName, MaxQDepth, MaxMsgLength, BackoutThreshold.*
  - *BackoutRequeueName, Shareability, DefInputOpenOption, HardenGetBackout.*
  - *MsgDeliverySequence, RetentionInterval, DistLists, Usage.*

© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

– *InitiationQName, TriggerControl, TriggerType, TriggerMsgPriority.*

– *TriggerDepth, TriggerData, DefinitionType, QDepthHighLimit.*

– *QDepthLowLimit, QDepthMaxEvent, QDepthHighEvent QDepthLowEvent,QServiceInterval,QServiceIntervalEvent.*

Since we need to change the input queue to become *GetInhibited* we should configure a PCF command message with the following:

- A PCF header indicating the change queue command.
- The QName and Qtype parameters.
- An optional parameter for the local queue: *InhibitGet.*

The parameters must occur in the following order:

1 All required parameters, in the specified order.

2 Optional parameters as required, in any order, unless specifically noted in the PCF definition.

Defining the PCF command message

MQSI V2.01 does not include the parser for PCF commands but it does come with a PCF header type, MQCFH structure, pre-defined to the system. So we just need to define the following:

- The format of the PCF integer parameter *MQCFIN*.
- The format of the PCF string parameter *MQCFST.*
- The message repository of the MQSI.

We can accomplish this by simply importing the above parameter structures into MQSI. Depending on the installation, these c structures can usually be found in *C:/Program Files/MQSeries/Tools/c/include/mqcfc.h.*

We need to cut and paste from the *mqcfc.h* the sections for *MQCFIN* and *MQCFST*. We will also need to do some editing before the import can be successful.

The following header files should be included after the statements for

© 2001. Reproduction prohibited. Please inform Xephon of any infringement.

typedef MQLONG and MQCHAR.

```
/*  MQCFIN Structure -- PCF Integer Parameter                          */
 typedef long MQLONG;
 typedef struct tagMQCFIN {
   MQLONG  Type;         /* Structure type */
   MQLONG  StrucLength;  /* Structure length */
   MQLONG  Parameter;    /* Parameter identifier */
   MQLONG  Value;        /* Parameter value */
  } MQCFIN;
/*  MQCFST Structure -- PCF String Parameter                           */
 typedef long MQLONG;
 typedef char MQCHAR;
 typedef struct tagMQCFST {
   MQLONG  Type;            /* Structure type */
   MQLONG  StrucLength;     /* Structure length */
   MQLONG  Parameter;       /* Parameter identifier */
   MQLONG  CodedCharSetId;  /* Coded character set identifier */
   MQLONG  StringLength;    /* Length of string */
   MQCHAR  String[1];       /* String value - first character */
  } MQCFST;
```

Having completed the above editing we can now define a new MessageSet called *MQCFH1* (note that MQCFH is predefined by MQSI) and start importing the header files into MQSI.

Once we have successfully imported the header files we can start defining the message for the PCF command. Please refer to the MQSI manual for details on creating messages, but note that the sequence is:

- Create elements.
- Create message type.
- Create message base on the message type.

Since we already have the *MQCHF_TYPE* (MQSI pre-defined) and *MQCFST_TYPE* and *MQCFIN_TYPE* (from a previous import) we need to define four elements:

- *PCF_HDR* of type *MQCHF_TYPE*.
- *PCF_PARM1* of type *MQCFST_TYPE*.
- *PCF_PARM2* of type *MQCFIN_TYPE*.
- *PCF_PARM3* of type *MQCFIN_TYPE*.

Then we can move on to define a message type *PCF_CMD_TYPE* that is composed of the above elements.

© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

Finally, we can create the message *PCF_CMD* base on the *PCF_CMD_TYPE*.

COMPUTE NODE THAT ISSUES THE PCF COMMAND

The basic function of this compute node is to configure the header and parameters of the PCF command that will change the attribute of the input queue to *GetInhibited*.

First of all, we need to get the input queue header information. From this we can get the name of the queue we need to change to *GetInhibit*, so we check the copy message header box.

Secondly, we need to set the output message to the **PCF_CMD** message that we defined earlier, with the ESQL statements detailed below.

```
SET OutputRoot.Properties.MessageSet = 'DJOQ9C8Ø7EØØ1';
SET OutputRoot.Properties.MessageType = 'PCF_CMD';
SET OutputRoot.Properties.MessageFormat = 'CWF';
```

The **PCF_CMD** will be a request message of format *MQADMIN* and it will expect the **PCF_CDM** response to go to the *ReplyToQ name*, which, in this case, is *ALEX_TEST_OUT*.

```
SET OutputRoot.MQMD.MsgType = 1;
SET OutputRoot.MQMD.Feedback = Ø;
SET OutputRoot.MQMD.Format = 'MQADMIN';
SET OutputRoot.MQMD.ReplyToQ = 'ALEX_TEST.OUT';
```

The subsequent ESQLs will propagate the PCF command header and the three parameters that follow. The comments on the ESQL are self-explanatory.

```
-- Set the PCF Header
-- change queue to get-inhibit
SET OutputRoot.MRM.PCF_HDR.Type = 1;
SET OutputRoot.MRM.PCF_HDR.StrucLength = 36;
SET OutputRoot.MRM.PCF_HDR.Version = 1;
-- Set the PCF command to MQCMD_CHANGE_Q = 8L
SET OutputRoot.MRM.PCF_HDR.Command = 8;
SET OutputRoot.MRM.PCF_HDR.MsgSeqNumber = 1;
SET OutputRoot.MRM.PCF_HDR.Control = 1;
SET OutputRoot.MRM.PCF_HDR.CompCode = Ø;
SET OutputRoot.MRM.PCF_HDR.Reason = Ø;
-- there will be 3 parameter follows: 2 required (QName, Qtype) and 1
optional (InhibitGet)
```

© 2001. Reproduction prohibited. Please inform Xephon of any infringement.

```
SET OutputRoot.MRM.PCF_HDR.ParameterCount = 3;
-- First parameter, the QName whose property is to be Changed
-- set the type to MQCFT_STRING = 4L
SET OutputRoot.MRM.PCF_PARM1.Type = 4;
-- set the length = MQCFST_STRU_LENGTH_FIXED = 20L + MQ_Q_NAME_LENGTH =
48L
SET OutputRoot.MRM.PCF_PARM1.StrucLength = 68;
-- set the parameter QName to MQCA_Q_NAME = 2016L
SET OutputRoot.MRM.PCF_PARM1.Parameter = 2016;
SET OutputRoot.MRM.PCF_PARM1.CodedCharSetId = 0;
-- get this QName from the Input Message Header into this node
SET OutputRoot.MRM.PCF_PARM1.StringLength =
CAST(LENGTH(InputRoot.MQMD.SourceQueue) AS INT);
SET OutputRoot.MRM.PCF_PARM1.String = InputRoot.MQMD.SourceQueue;
-- Second parameter is the QType
-- set the type to MQCFT_INTEGER = 3L
SET OutputRoot.MRM.PCF_PARM2.Type = 3;
-- the Length of this integer parameter is MQFIN_STRUC_LENGTH = 16L
SET OutputRoot.MRM.PCF_PARM2.StrucLength = 16;
-- Set QType to MQIA_Q_TYPE = 20L
SET OutputRoot.MRM.PCF_PARM2.Parameter = 20;
-- Set MQQT_LOCAL = 1L
SET OutputRoot.MRM.PCF_PARM2.Value = 1;
-- Thrid parameter is the Option: to set the GetInhibit
-- set the type to MQCFT_INTEGER = 3L
SET OutputRoot.MRM.PCF_PARM3.Type = 3;
-- the Length of this integer parameter is MQFIN_STRUC_LENGTH = 16L
SET OutputRoot.MRM.PCF_PARM3.StrucLength = 16;
-- Set InhibitGet MQIA_INHIBIT_GET = 9L
SET OutputRoot.MRM.PCF_PARM3.Parameter = 9;
-- Set MQQA_GET_INHIBITED = 1L
SET OutputRoot.MRM.PCF_PARM3.Value = 1;
```

This **PCF_CMD** message is then put to the *SYSTEM.ADMIN.COMMAND.QUEUE* for execution.

THE *STOPMESSAGEFLOW* FLOW

To test out the **PCF_CMD** command we can build a very simple message flow as shown in Figure 4 – the trace nodes are there only to verify that the command message is configured properly.

For simplicity, the MQInput node was configured to point to an input queue, *ALEX_TEST_IN*, and the message domain was XML.

In order to test the message flow, the message set *MQCFH1* created earlier must be deployed to the broker. Once the message flow is deployed to the execution group any XML message sent to the input

© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

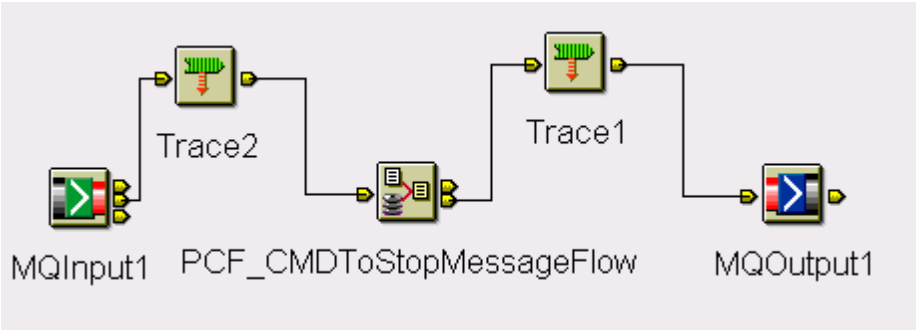

*Figure 4: Testing the **PCF_CMD** command*

queue will set the P**CF_CMD** in action and the input queue will become *GetInhibited*.

The trace of the PCF command will be as shown below.

PCF COMMAND TRACE

```
*** 2001-06-23 17:25:50.013
(
  (0x1000000)Properties = (
    (0x3000000)MessageSet     = 'DJOQ9C807E001'
    (0x3000000)MessageType    = 'PCF_CMD'
    (0x3000000)MessageFormat  = 'CWF'
    (0x3000000)Encoding       = 546
    (0x3000000)CodedCharSetId = 437
    (0x3000000)Transactional  = TRUE
    (0x3000000)Persistence    = FALSE
    (0x3000000)CreationTime   = GMTTIMESTAMP '2001-06-24 00:25:49.480'
    (0x3000000)ExpirationTime = GMTTIMESTAMP '2001-06-24
00:25:49.862998'
    (0x3000000)Priority       = 0
    (0x3000000)Topic          = NULL
  )
  (0x1000000)MQMD       = (
    (0x3000000)SourceQueue      = 'ALEX_TEST_IN'
    (0x3000000)Transactional    = TRUE
    (0x3000000)Encoding         = 546
    (0x3000000)CodedCharSetId   = 437
    (0x3000000)Format           = 'MQADMIN'
    (0x3000000)Version          = 2
    (0x3000000)Report           = 0
    (0x3000000)MsgType          = 1
    (0x3000000)Expiry           = GMTTIMESTAMP '2001-06-24
00:25:49.862998'
```

© 2001. Reproduction prohibited. Please inform Xephon of any infringement.

```
    (0x3000000)Feedback         = 0
    (0x3000000)Priority         = 0
    (0x3000000)Persistence      = 0
    (0x3000000)MsgId            =
X'414d51204d5153492020202020202020fdf343b93500000'
    (0x3000000)CorrelId         =
X'000000000000000000000000000000000000000000000000'
    (0x3000000)BackoutCount     = 0
    (0x3000000)ReplyToQ         = 'ALEX_TEST_OUT'
    (0x3000000)ReplyToQMgr      = 'MQSI
'
    (0x3000000)UserIdentifier   = 'Alexau      '
    (0x3000000)AccountingToken  =
X'16010515000000e551800cb8a4a40ae558d40aea030000000000000000000000b'
    (0x3000000)ApplIdentityData = '                                '
    (0x3000000)PutApplType      = 11
    (0x3000000)PutApplName      = 'C:\WINNT\System32\MMC.EXE   '
    (0x3000000)PutDate          = DATE '2001-06-24'
    (0x3000000)PutTime          = GMTTIME '00:25:49.480'
    (0x3000000)ApplOriginData   = '    '
    (0x3000000)GroupId          =
X'000000000000000000000000000000000000000000000000'
    (0x3000000)MsgSeqNumber     = 1
    (0x3000000)Offset           = 0
    (0x3000000)MsgFlags         = 0
    (0x3000000)OriginalLength   = 12
  )
  (0x1000008)MRM        = (
    (0x1000000)PCF_HDR   = (
      (0x3000000)Type           = 1
      (0x3000000)StrucLength    = 36
      (0x3000000)Version        = 1
      (0x3000000)Command        = 8
      (0x3000000)MsgSeqNumber   = 1
      (0x3000000)Control        = 1
      (0x3000000)CompCode       = 0
      (0x3000000)Reason         = 0
      (0x3000000)ParameterCount = 3
    )
    (0x1000000)PCF_PARM1 = (
      (0x3000000)Type           = 4
      (0x3000000)StrucLength    = 68
      (0x3000000)Parameter      = 2016
      (0x3000000)CodedCharSetId = 0
      (0x3000000)StringLength   = 12
      (0x3000000)String         = 'ALEX_TEST_IN'
    )
    (0x1000000)PCF_PARM2 = (
      (0x3000000)Type        = 3
      (0x3000000)StrucLength = 16
      (0x3000000)Parameter   = 20
      (0x3000000)Value       = 1
```


© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

```
    )
    (0x1000000)PCF_PARM3 = (
      (0x3000000)Type        = 3
      (0x3000000)StrucLength = 16
      (0x3000000)Parameter   = 9
      (0x3000000)Value       = 1
    )
  )
)
```

THE EXCEPTION FLOW

The exception processing message flow can now be incorporated with the PCF command node to 'stop' the message flow when a system error occurs (see Figure 5).

The *WriteErrorMsg* compute node is updated to include the SQL code generated if a *DatabaseException* occurred. The SQL code will be written out to the error message if it was a *DatabaseException*. The revised ESQL code would be as shown below.

REVISED ESQL CODE

```
DECLARE I INTEGER;
SET I = 1;
```

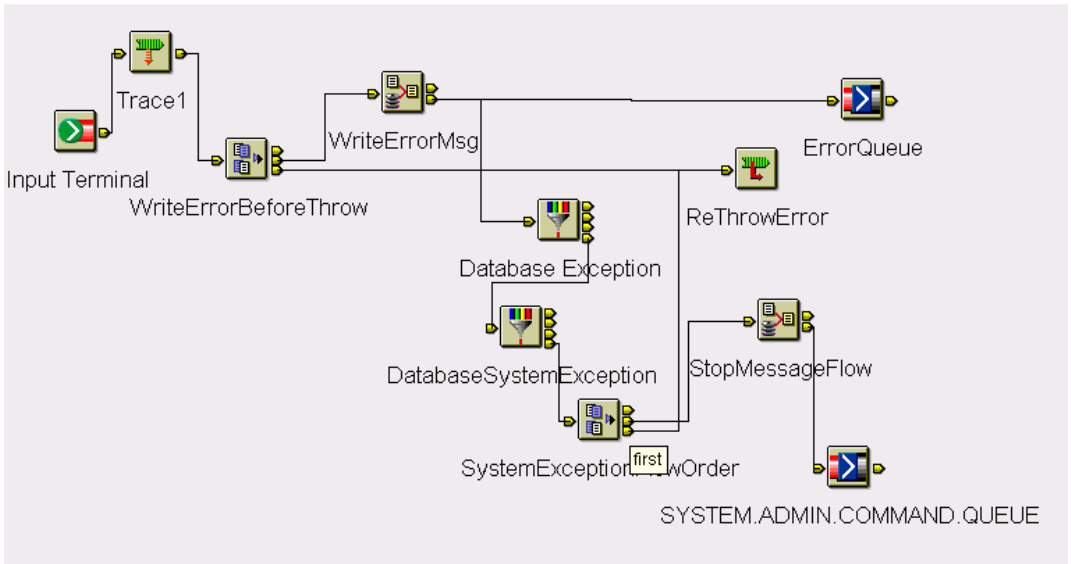

Figure 5: Stopping the message flow

© 2001. Reproduction prohibited. Please inform Xephon of any infringement.

```
WHILE I < CARDINALITY(InputRoot.*[]) DO
     SET OutputRoot.*[I] = InputRoot.*[I];
     SET I=I+1;
END WHILE;
-- Enter SQL below this line.  SQL above this line might be regenerated,
causing any modifications to be lost.
/* Error number extracted from exception list */
DECLARE Error INTEGER;
/* Current path within the exception list */
DECLARE Path CHARACTER;
DECLARE ErrorType  CHARACTER;
DECLARE ErrorTrans CHARACTER;
DECLARE ErrorText  CHARACTER;
DECLARE SQLCode CHARACTER;
SET ErrorType  = '';
SET ErrorTrans = '';
SET ErrorText  = '';
/* Start at first child of exception list */
SET Path ='InputExceptionList.*[1]';
/* Loop until no more children */
WHILE EVAL( 'FIELDNAME(' || Path || ') IS NOT NULL' ) DO
  /* Check if error number is available */
  IF EVAL( 'FIELDNAME(' || Path || '.Number) IS NOT NULL' ) THEN
    /* Remember only the deepest error number */
    SET Error    = EVAL( Path || '.Number' );
    SET ErrorType  = EVAL('FIELDNAME('||Path||')');
     IF ErrorType = 'DatabaseException' THEN
      SET SQLCode = COALESCE(EVAL(Path || '.Insert[2].Text'),'');
     END IF;
    SET ErrorTrans = EVAL(Path || '.Label');
    SET ErrorText  = EVAL(Path || '.Text');
    IF ErrorType = 'UserException' THEN
     SET ErrorText = ErrorText || ' (' || (EVAL(Path || '.Insert.Text'))
|| ')';
    END IF;
  END IF;
  /* Step to last child of current element (usually a nested exception
list) */
  SET Path = Path || '.*[LAST]';
END WHILE; /* End loop */
SET OutputRoot.XML.Exception.Type     = ErrorType;
IF ErrorType = 'DatabaseException' THEN
  SET OutputRoot.XML.SQLCode          = SQLCode;
END IF;
SET OutputRoot.XML.Exception.Number       = CAST(Error AS CHAR);
SET OutputRoot.XML.Exception.SuspenseTimestamp = CAST(CURRENT_TIMESTAMP
AS CHAR);
SET OutputRoot.XML.Exception.InboundQueue = InputRoot.MQMD.SourceQueue;
SET OutputRoot.XML.Exception.Transaction  = ErrorTrans;
SET OutputRoot.XML.Exception.Reason       = ErrorText;
```

 © 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

```
SET OutputRoot.XML.Exception.DetailException   = InputExceptionList;
```

The Database Exception filter node will check for *DatabaseException* first, if the condition was true, and will then check for the common DB2 system error:

```
-- DB2 System Error exceptions:
Root.XML.SQLCode IN('-922','-923','-924')
/*****************************************/
/* -922 = AUTHORIZATION FAILURE          */
/* -923 = CONNECTION NOT ESTABLISHED     */
/* -924 = DB2 CONNECTION INTERNAL ERROR  */
```

If the condition for a DB2 system error was true it will then execute the *StopMessageFlow* compute node and issue the PCF command to the *SYSTEM.ADMIN.COMMAND.QUEUE*, and then re-throw the exception so that the message will be rolled back to the BackOut queue.

The key to success in putting a message into the *SYSTEM.ADMIN.COMMAND.QUEUE* is, in the MQOutput node, to select No for the Transaction Mode drop down list in the Advanced tab, so that the PCF message written is committed before we re-throw the exception.

CONCLUSION

One key piece of advice when coding the exception processing sub-flow is to avoid manipulating the message body content. This is to avoid any parser exception being thrown from the exception processing sub-flow by the system.

A message parser will be called if we manipulate the body content of the message, so if the exception itself was a parser exception before it was thrown to the exception processing sub-flow, the system will re-throw a parser exception again and affect the ability of the sub-flow to generate an error message to the error queue.

The above sample is by no means complete, but will serve well with the datagram type of message. There is room for improvement for the request/reply message type. For example, when the message type is a request, and it encounters some exception during the hub processing, instead of writing an error message to the error queue, you can write the error message back to the *ReplyToQ* specified in the *MQMD* of the

© 2001. Reproduction prohibited. Please inform Xephon of any infringement.

request message, so that the requesting program gets not only a *timeout error*, but also a more descriptive error message and can, therefore, take appropriate follow-up action.

Alex Au
I/T Architect, IBM Global Services (USA) © Alex Au

The CSQUTIL utility

The CSQUTIL is a utility provided with MQSeries to help issue commands, perform backup and restore, and reorganize tasks.

MAKEDEF COMMAND

In this example, MQSeries commands are passed from an input dataset referenced by DDname *REBUILD* to Queue Manager *MQM1* on the OS/390 platform in batch. A list of DEFINE statements is produced that describes the objects in this MQSeries subsystem. Any changes or new definitions are encompassed and the statements are used to regenerate all or part of MQM1's objects and storage classes.

```
//USERID JOB (SØ9P91T),'DAVIES',CLASS=T,MSGCLASS=B,
//    NOTIFY=&USERID
//*  PROGRAM CSQUTIL ISSUES COMMANDS TO QMGR IDENTIFIED BY
//*  PARM='NNNN' ON THE EXEC CARD
//*  MESSAGES ARE PRINTED TO DD SYSPRINT
//*  REBUILD AN MQSERIES SUBSYSTEMS USER DEFINITIONS
//COPYDEFS EXEC PGM=CSQUTIL,PARM='MQM1'
//STEPLIB  DD DISP=SHR,DSN=SYS1.SCSQANLE
//                  DD DISP=SHR,DSN=SYS1.SCSQAUTH
//REBUILD  DD DISP=SHR,DSN=MQM1.REBUILD.DATASET(DEFMQM1)
//SYSPRINT DD SYSOUT=*
//SYSIN        DD *
COMMAND DDNAME(REBUILD)
DEFMQM1 sample definitions
DISPLAY STGCLASS(*) ALL
DISPLAY QUEUE(*) ALL
DISPLAY NAMELIST(*) ALL
DISPLAY PROCESS(*) ALL
DISPLAY CHANNEL(*) ALL
```

Saida Davies
IBM (UK) © S Davies

 © 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

HIS and its MSMQ MQSeries Bridge part 2: deployment and configuration

In the first of this three-part series of articles on HIS (Host Integration Server) and the MSMQ to MQ Bridge we examined the ways in which MSMQ and MQ can be linked, with a specific emphasis on the planning stages. This month, we will focus on deployment and configuration.

BRIDGE SETUP REQUIREMENTS

In this section we will examine the steps required to set up the Bridge:

- Basic configuration.
- Computers involved.
- Information required.

Basic configuration

The basic configuration allows MSMQ and MQSeries applications to converse with one another, and can consist of:

- One Bridge, installed on a Win2K server or NT 4.0 Enterprise server.
- One MSMQ server installed on the same computer as the Bridge – the MSMQ Routing Server in NT4 Enterprise server or MSMQ server with routing in Win2K.
- MQSeries Client for NT and the Bridge on the same computer.
- One MQSeries queue manager installed on a supported computer.

Computers involved

You need to work on two computers when installing the Bridge:

- The MSMQ and Bridge computer.
- The computer where MQSeries queue manager is installed; sometimes called the MQSeries computer.

Data transfers must be performed on each computer.

© 2001. Reproduction prohibited. Please inform Xephon of any infringement.

Information required

The following items of information, along with their respective values, are required and must be gathered prior to the configuration process.

- Machine name (ie *MSBRIDGE1*).
- Directory in which the MQSeries Client for Win2K or NT is installed (ie *D:\MQCLIENT*).

The data required for the MQSeries computer must all be in upper case, and is as follows:

- Connection data
  - when using TCP/IP, the computer name or IP address (ie *IBMNT* or 192.10.10.5) and the port number (ie 1414).
  - for SNA, the LU 6.2 Side Information Record (CPI-C Symbolic Destination Name).
- Name of the MQSeries Queue Manager (ie *IBMNT*).

BRIDGE PREREQUISITES

Your system should meet the requirements detailed below.

Bridge platforms

- The server can be installed on any Win2K server or NT 4.0 Enterprise Edition server.
- The Administrator Client can be installed on Win2K Professional, Win2K server with Terminal Services installed, NT 4.0 Workstation Service Pack 6a, or NT 4.0 Server Terminal Server Edition with Service Pack 6.

Prerequisites for NT 4.0 machines

The software detailed below should already be installed on the NT 4.0 computer where you will install the Bridge or Administrator Client.

Server prerequisites

- Windows NT Server 4.0 Enterprise Edition.

© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

- MSMQ 1.0 (NT 4.0 Option Pack) set up as server (PEC, PSC, BSC, or Routing Server).
- IBM MQSeries Client for NT or IBM MQSeries for NT (with server and client installed).
- TCP/IP or SNA (LU 6.2) link to an MQSeries Queue Manager (QM).

Administrator client prerequisites

- NT 4.0 Workstation with Service Pack 6a, or NT 4.0 Server Terminal Server Edition with Service Pack 6.
- Any configuration of MSMQ 1.0.

Prerequisites for Windows 2000 machines

The software detailed below should already be installed on the computer where you will install the Bridge Server or Administrator Client.

Server prerequisites

- Win2K server.
- MSMQ server, but not part of a workgroup, with Routing Enabled.
- IBM MQSeries Client for NT or IBM MQSeries for NT (with server and client installed).
- TCP/IP or SNA (LU 6.2) link to an MQSeries Queue Manager (QM).

Administrator Client prerequisites

- Win2K Professional, or Win2K server with Terminal Service.
- MSMQ set up, but not part of a workgroup.

Prerequisites for the MQSeries machine

- For OS/390 systems, make sure that your MQSeries Queue Manager is configured with the Client Attachment feature.

© 2001. Reproduction prohibited. Please inform Xephon of any infringement.

General tab

The General tab displays the following information:

- Path name: network name of the Bridge computer.
- Service: name of the Microsoft Bridge Service.
- Version: version of the Microsoft Bridge Service.
- Status: status of the Microsoft Bridge Service, ie running, paused, or stopped.

Advanced tab

The Bridge, by default, allocates one thread for each type of message pipe. If the Bridge is connected to more than one MQSeries queue manager you must allocate a larger number of threads, thereby improving performance, and if a pipe to one QM fails, the other pipes will continue running. For each type of message pipe, specify:

- Max threads: the maximum number of threads, with a recommendation of one per QM.
- Refresh queue cache: the interval in minutes that the message pipes check for cache timeout.
- Support MSMQ to Bridge encryption: check this option to enable the encryption feature from MSMQ to the Bridge.
- Replace '.' with '-' in the queue manager: MQSeries does not support 'hyphen' (-) characters, so two tasks are required:
  - the system administrator needs to replace the hyphen in an MSMQ computer definition in MQSeries with a period (.).
  - check the checkbox to enable the Bridge to replace a period (.) from the remote queue manager name on MQSeries with a hyphen (-).

MQI channels tab

Th Bridge accesses MQSeries via MQI channels defined in both MQSeries and the Bridge. Each MQI channel connects a Bridge to an

 © 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

MQSeries queue manager. You define one MQI channel for each connected network or foreign site. First, define the channels in the Bridge properties. Later, you export the definitions to MQSeries. In the dialog box, click Add for a new MQI channel, Properties to edit the settings for an existing channel, or Remove to delete a channel. On the General tab of the Channel Properties window, specify the following options:

- Channel Name: a legal MQSeries name for the MQI channel, such as the same name you assigned to the connected network representing MQSeries in MSMQ, ie *IBMNT_CN*.
- MQSeries Queue Manager: the MQSeries Queue Manager to which the channel connects.
- Transport Type: TCP/IP or SNA LU 6.2 communication. On the Address tab of the Channel Properties window, specify parameters for TCP/IP, such as IP Address, Port of the MQSeries listener; and for SNA LU 6.2, specify Side Information Record, which is the CPI-C Symbolic Name defined in HIS 2000.

On the Security tab of the Channel Properties window you can specify the MCA user, which is an existing or new MQSeries user name, for example *FMQUSER1*, under which the server side of the MQI channel runs. In MQSeries, you should set the permissions of the MCA user or queues that the Bridge can address. If you do not specify an MCA user, the server side of the channel runs under the default user name, which is the value of the MQSeries *SYSTEM.DEF.SVRCONN* parameter.

Working with a CN (Connected Network)

Before adding a CN you can add a CN to a Bridge:

- Identify the CN in MSMQ and associate the Bridge computer with the CN.
- Identify an MQI Channel for the Bridge.
- Open the Bridge Manager. To add a CN, right-click on the Bridge to which the CN is connected. Choose ‘New CN’ from the pop-up menu, and select the CN name from the list.

© 2001. Reproduction prohibited. Please inform Xephon of any infringement.

When adding a new CN, four message pipes are added to the console tree:

- MSMQ-MQS transactional: MSMQ to MQSeries transactional.
- MSMQ-MQS non-transactional: MSMQ to MQSeries non-transactional.
- MQS-MSMQ transactional: MQSeries to MSMQ transactional.
- MQS-MSMQ non-transactional: MQSeries to MSMQ non-transactional.

Right-click on the new CN and on each message pipe to set their properties. To delete a CN, right-click on the CN and select Delete from the pop-up menu. To set the properties of a CN, right-click its icon in the Bridge Manager and choose the Properties option.

On the General tab, in the CN properties window, you can specify the following:

- MQSeries QM name: select from the list of MQSeries queue managers that you have defined as MQI channels. If the same one is on two connected networks do not connect both of them to the same QM.
- Reply to QM name: the default MSMQ QM to which MQSeries should return report or acknowledgment messages. You enter the name of the Bridge computer, ie *XEPHONBridge1*. The Bridge Manager exports your entry to MQSeries as a queue manager alias. If the Windows computer name contains an invalid hyphen (-), replace it with another valid character, such as an underscore (_), in the Reply to QM Name field. If you want to receive acknowledgments by a non-transactional instead of transactional message pipe, add a % sign to the name (ie *XEPHONBridge1%*). The alias is used in MQSeries to identify the transmission queue for sending acknowledgments and you can redirect the acknowledgments by specifying a different alias. For example, if you want to receive acknowledgments on another computer, enter the name of the computer and define the name as an alias in MQSeries.
- Startup: enabled or disabled at the Bridge.

© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

Setting message pipe properties

To set the properties of a message pipe, right-click its icon in the Bridge Manager, and choose the Properties option. There are different intuitive icons for stopped, paused, pending, recovering, running, and error.

General tab: message pipe

The General tab of the Message Pipe Properties window displays:

- Status: running, paused, pending, recovering, stopped, or error.
- Startup: enabled or disabled at Bridge startup.

For the MQSeries to MSMQ transactional and non-transactional message pipes:

- Transmission Queue Name is a unique MQSeries transmission queue name for the pipe. The default names are *<CN name>.XMITQ* for the transactional message pipe and *<CN name>.XMITQ.HIGH* for the non-transactional message pipe. Different names can be specified. For example, if the Bridge computer name is *XEPHONBridge1*, the default transmission queue names are *XEPHONBridge1.XMITQ* and *XEPHONBridge1.XMITQ.HIGH*.

Batch tab

The Bridge batches messages, which results in a performance boost, so increasing the batch size can improve performance. However, with transactional message pipes, this can increase the quantity of retransmitted data after a communication failure. To change the batch size, specify:

- Max number of messages: the maximum number of messages in a batch.
- Max accumulated size: the maximum byte size of a batch.
- Max accumulated time: the maximum time in milliseconds during which messages are batched.

© 2001. Reproduction prohibited. Please inform Xephon of any infringement.

Cache tab

The Bridge caches queue handles and re-uses them when several messages are sent to the same queue, thereby reducing the overhead of opening and closing a queue for each message. You can specify:

- Cache expiry: the time in minutes after which the Bridge closes an unused queue handle.

Retry tab

When a message pipe fails, the Bridge tries to restart it automatically. You can specify:

- Count: maximum number of retries.
- Delay: interval between retries.

You can specify a short cycle of four retries at 40-second intervals; if the connection is still not successful the system continues in a long cycle of 12 retries at 400-second intervals. When using TCP/IP, set the delay to at least twice the 'keep alive' time of the destination MQSeries computer.

CHOOSE NAMES FOR MSMQ AND MQSERIES ENTITIES

You should choose names identical to or derived from other names that you recorded (to simplify the process) for certain MSMQ and MQSeries entities that you will identify during the configuration process. MSMQ names and values are required for the following:

- A connected network or foreign site (ie *IBMNT_CN*).
- The queue manager on the foreign computer (ie *IBMQM*).
- A foreign queue (ie *IBM.NT.QUEUE*).
- An MSMQ queue to which you will send test messages (ie *MSBRDIGE1.QUEUE*).

MQSeries names and values are required for the following:

- MQI channel (ie *IBMNT_CN*).
- Transmission queue for normal service (ie *MSBRIDGE1.XMITQ*).

© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

- Transmission queue for high service (ie *MSBRIDGE1.XMITQ.HIGH*).
- An MQSeries queue to which you will send test messages (ie *IBM.NT.QUEUE*).

INSTALLING AND CONFIGURING THE BRIDGE

Installing the Bridge software

The procedure is the same for installing either the Server or the Administrator Client. Choosing Server installs the Bridge and the Bridge Manager; choosing Administrator Client installs the Bridge Manager.

To install the Bridge

- Insert the HIS 2000 CD-ROM and allow the startup page to appear.
- Click Install Server or Install Administrator Client.

When using the TCP/IP protocol ensure that the setting in the MQSeries Queue Manager initialization file (*qm.ini*) Keep Alive is set to 'Yes' and the Keep Alive Time is half or less of the message pipes' retry delay, allowing the MQSeries Listener to release the resources of a broken connection before the Bridge retries the connection. To get to the message pipes retry delay, right-click a message pipe in the Bridge Manager, select Properties, and then select the Cache tab.

Configuring an MQI channel to use the SNA LU 6.2 protocol has been discussed earlier in this article.

Configuring the Bridge on Win2K

Constructing the Bridge on Win2K requires an ordered three-stage process, which requires MSMQ 2.0 to be configured first, followed by the Bridge, and then MQSeries. Configuring MSMQ 2.0 on Win2K is the only stage that is different from the configuration process on NT 4.0.

MSMQ 2.0 and the Bridge

Win2K comes with MSMQ 2.0, and provides additional functionality, including:

© 2001. Reproduction prohibited. Please inform Xephon of any infringement.

- Integration with Active Directory, removing the need to use a separate SQL server to maintain the MQIS.
- Mixed-mode operation, enabling MSMQ 1.0 and MSMQ 2.0 environments to co-exist.
- Performance improvements, particularly in the area of transactions.
- Workgroup Mode, enabling Win2K computers to use MSMQ 2.0 without the need for an Active Directory.

The Bridge is capable of running on Win2K and can be installed on any of the Win2K servers.

To use the Bridge on Win2K, install:

- Win2K server.
- MSMQ server (not in a workgroup) with Routing Enabled.
- IBM MQSeries Client for NT, Version 2.0, 5.0, or 5.1, or IBM MQSeries for NT, Version 2.0, 5.0, or 5.1, both Server and Client.
- The Bridge from HIS.

You should be aware of the changes in MSMQ 2.0 that are related to the Bridge, such as:

- In keeping with Active Directory terminology, the term 'Foreign Connected Network' has been replaced with 'Foreign Site', so a CN refers to a foreign site in Win2K.
- With changes in the MSMQ 2.0 COM API, many more of the MSMQ message properties are now exposed through the MSMQ COM API, such as the extension property (*PROPID_M_EXTENSION*), which is now accessible from Visual Basic, making it easier to override Bridge conversions.
- The creation of foreign sites and foreign computers is achieved using Active Directory Sites and Services.

Configuring MSMQ 2.0 on Win2K

Using Control Panel, select Add/Remove Components and install the Message Queueing Services and Enable Routing during the installation

© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

of MSMQ 2.0. This will allow you to define a foreign site and define routing links to the foreign site. This is required to set up a Bridge. MSMQ 2.0 should have been installed with Routing Enabled on a Win2K server. You will use this same server to install the MSMQ-MQSeries Bridge.

With the Win2K administrators utility Active Directory Users and Computers set to View Advanced Features, you will see the MSMQ object under the path: *Domain Controllers \ Your computer name \ MSMQ*. You can use Active Directory to configure MSMQ 2.0 for use with the MSMQ-MQSeries Bridge, but, by default, only Enterprise administrators can make changes. You can, however, make changes to the permissions to expand the users who have this capability. Win2K doesn't permit a period (.) character in computer names so replace it with a dash (-) and the MSMQ-MQSeries Bridge will map the dash back to a period when routing the message to its MQSeries destination.

To define a foreign site in Win2K

- Using Active Directory Sites and Services, select View, and Show Services Node.
- Under Services, right-click *MsmqServices* and select New Foreign Site.
- Enter the name for the foreign site, which is the name of the Foreign Connected Network in MSMQ 1.0.

To add a foreign computer representing MQSeries in Win2K

- Using Active Directory Sites and Services, select View, and Show Services Node.
- Under Services, right-click *MsmqServices* and select New Foreign Computer.
- Enter the name for the foreign computer. This name must be the same as that of the MQSeries Queue Manager.
- Select the name for the foreign site, and click OK.

© 2001. Reproduction prohibited. Please inform Xephon of any infringement.

To set the foreign site permission in Win2K

- Using Active Directory Sites and Services, expand the Sites folder, and select the Foreign Site created earlier.
- Right-click, and then select Properties.
- Select the Security tab, and select Everyone. If the Bridge was installed using a local account, enable Open Connector Queue; otherwise, select the account name used during setup.

To create a foreign queue in Windows 2000 and define the MQSeries queues

- Using Active Directory Users and Computers, select View, and Advanced Features and then View, and Users, Groups, and Computers As Containers.
- Expand the Computers folder and select the foreign computer that was just created.
- Right-click the MSMQ object and select New, followed by MSMQ Queue.
- Type the name for the queue and click OK. The name must be the same as that of the MQSeries queue.
- If you want to create a queue to accept transactional messages, type a name, select the Transactional check box, and then click OK.

A routing link is required to enable MSMQ 2.0 to route messages between the current Win2K site and the newly created foreign site that is used for MQSeries. To create a routing link:

- Using Active Directory Sites and Services, select View, and Show Services Node.
- Right-click *MsmqServices*, and select New, followed by MSMQ Routing Link.
- Set site one to the name of the foreign site that was just created. Set site two to the name of the current Win2K site.
- Set the routing link cost to ‘one’ and then click OK. Using a value greater than one is only relevant where multiple routing links are defined between sites and you want to enforce one route over

© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

another. Do not set this value to zero as it will cause routing of MSMQ messages to the foreign site to fail.

A site gate is an MSMQ server that is configured to route messages between sites on behalf of other clients and is the computer that will run the Bridge using the routing link that was just created. To define a site gate follow the steps detailed below:

- Using Active Directory Sites and Services, select View and Show Services Node.
- Under Services, right-click *MsmqServices;* the routing link that was just created should appear in the details panel.
- Right-click the routing link and select Properties.
- Select the Site Gates tab.
- In Site Servers, select the name of the computer that will run the Bridge, and then click Add.

The MSMQ Server that is on the same Win2K needs to be added to the foreign site created earlier.

- Using Active Directory Users and Computers, select View, and Advanced Features, followed by View and Users, Groups, and Computers as containers.
- Expand the Domain Controllers or Computers / *<Your server name> / MSMQ* folder.
- Right-click the MSMQ object and select Properties.
- Select the Sites tab.
- Select the foreign site created earlier, and then click Add to add this server to the Foreign Site.

Configuring MSMQ 1.0 on NT 4.0

To define a connected network in NT 4.0

- Open the MSMQ Explorer.
- Right-click the Enterprise icon. Point to New, and then click CN.
- Enter a name for the connected network.

© 2001. Reproduction prohibited. Please inform Xephon of any infringement.

- For the Protocol, select Foreign and click OK.

To add a foreign computer representing MQSeries in Windows NT 4.0

- In the MSMQ Explorer, expand the Sites folder to display the site where the Windows computer is located.
- Right-click the Sites icon. Point to New and click Foreign Computer.
- Copy the foreign computer name.
- Select the connected network name.
- Click Add and then click OK.

To connect to the Bridge in NT 4.0

- In the MSMQ Explorer, right-click the Windows computer icon.
- Click Properties and then click the Network tab. Click Add.
- In the Connected Network Name dialog box, select the connected network name.
- Click OK three times. The reboot warning is displayed, but do not reboot at this time.
- Using Control Panel Services, stop and restart the Microsoft Message Queue Server. This step is equivalent to the reboot required from the previous step.

To set the Connected Network permission in NT 4.0

- Expand the Connected Networks folder in MSMQ Explorer.
- Right-click the connected network.
- Click Properties and then click the Security tab. Click Permissions.
- Set Type of Access for the Everyone group. If the Bridge was installed using a local account, set Type of Access to Special, otherwise select the account name used during setup.
- Select the Open Connector check box.
- Click OK three times. Your PSC and BSC computers may report

© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

an ‘Update Not Available Immediately’ message. Click OK to bypass these warnings.

To create a foreign queue in NT 4.0

- Expand the Sites folder.
- Right-click the foreign computer.
- Select New and then select Queue.
- Enter the foreign queue name and click OK. The complete name is required, ie *XEPHONBRIDGE1.QUEUE*.

Configuring the MSMQ-MQSeries Bridge Manager

To define an MQI channel

- On the Windows computer, open the Bridge Manager.
- Expand the Enterprise icon.
- Right-click the Bridge Service icon. Click Stop, and then click OK.
- Right-click the Bridge Service icon, and click Properties.
- On the MQI Channels tab, click Add.
- Enter the Channel Name.
- Enter the Queue Manager Name.
- For Transport Type, select TCP/IP or SNA LU 6.2.
- On the Address tab, if you connect by TCP/IP, specify the IP address or computer name to Address and the port number to Port. For SNA LU 6.2, specify the LU 6.2 Side Information Record.
- Click OK twice. A message appears warning you about changing the MQI Channel configuration. For now, click OK to bypass this message.

To add the CN

- Right-click the Bridge Service icon in the Bridge Manager. Point to New and then click CN.

© 2001. Reproduction prohibited. Please inform Xephon of any infringement.

- Select CN from the drop-down list box.
- Click OK. The CN Properties window (General tab) appears.
- Select the MQSeries QM Name.
- For the Reply to QM Name, enter the name of the Windows computer.
- For Startup, select Enabled and click OK.
- The Bridge Manager displays four message pipes (MSMQ-MQS transactional, MSMQ-MQS, etc). The message pipes are auto-started by default. You can disable the auto-start option in the appropriate Properties page.

To disable the message pipes

- In the Bridge Manager, right-click the first message pipe icon, MSMQ-MQS transactional, and then click Properties.
- On the General tab, select Disabled and click OK.
- Repeat the steps for the second message pipe, MSMQ-MQS, and so on.

To export an MQSeries Server Definition file

- Right-click the Bridge Service icon in the Bridge Manager, and then click Export Server Definitions.
- Enter a directory name to save the definition file (the default is *C:\Program Files\Host Integration Server\MQBridge)* and then click OK. The Manager saves the file with an extension of *.TXT*. If you define more than one CN, a definition file for each MQSeries Queue Manager is created in the directory.
- Transfer the file(s) to the MQSeries computer. If you use FTP to transfer the file, be sure to specify the ASCII transfer option.

To export an MQSeries Client Definition file

- Right-click the Bridge Service icon and then click Export Client Definitions.
- Enter the directory name to save the definition file (the default is

© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

C:\Program Files\Host Integration Server\MQBridge) and then click OK. The Manager saves the file with the name *ClientDf.txt*.

- Transfer the file to the MQSeries computer. If you transfer by FTP, be sure to specify the ASCII transfer option.

Configuring MQSeries

To run the MQSeries Server Definition file

- On the MQSeries computer, run the MQSeries Server definition file that you transferred from the Bridge Manager.
- At the command prompt, run the MQSC command, such as **RUNMQSC IBMNT < IBMNT.TXT > SERVREPORT.OUT** (substitute the MQSeries Queue Manager name for *IBMNT*).

The MQSeries Queue Manager is now configured. Review the *SERVREPORT.OUT* file to be sure that the definitions ran successfully.

To run the MQSeries Client Definition file

Still on the MQSeries Queue Manager computer, run the MQSeries Client Definition file that you transferred from the MSMQ-MQSeries Bridge Manager.

- At the command prompt, run the MQSC command, ie: **>RUNMQSC IBMNT < ClientDf.txt > CLIENTREPORT.OUT**.
- The MQSC command creates a channel file called *AMQCLCHL.TAB*, located in the directory *MQMDirectory/QMGRS/<QueueManagerName>/@IPCC* (the exact location may differ on various platforms).
- Review the *CLIENTREPORT.OUT* file to be sure that the definitions ran successfully.
- Transfer the *AMQCLCHL.TAB* file to the MQSeries Client directory on the Windows computer. If you transfer by FTP, be sure to specify the binary transfer option.

To configure the MQSeries Client

- To complete the MQSeries configuration, return to the Windows computer where the Bridge is installed.

© 2001. Reproduction prohibited. Please inform Xephon of any infringement.

- On NT 4.0, right-click the My Computer icon, click Properties, and then select the Environment tab. Using Win2K, right-click the My Computer icon from your desktop, click Properties, select Advanced tab, and then click on Environment Variable.
- Check that the following variables are in the System Variables. If not, set them correctly.
  - *MQCHLLIB* should be the directory path of the channel file (by default, the MQSeries Client directory).
  - *MQCHLTAB* should be the file name of the channel file (by default, A*MQCLCHL.TAB*).
  - Check that the environment variable *MQSERVER* is not defined.
  - Click OK. On Windows NT 4.0, reboot the computer.

TESTING THE INSTALLATION

Conveniently, test programs are provided in the directory *Drive:\Program Files\Host Integration Server\System*. Testing involves the following steps:

- Creating the test queues.
- Testing the MQSeries Client definitions.
- Starting the MSMQ-MQSeries Bridge.
- Sending test messages from MSMQ to MQSeries.
- Sending test messages from MQSeries to MSMQ.

To create the test queues

You can use queues that already exist on the Windows and MQSeries computers or create test queues. To create a new MSMQ queue on NT 4.0:

- Open the MSMQ Explorer.
- Right-click the Windows computer where the Bridge is installed, point to New, and then click Queue.
- Enter the queue name and click OK.

© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

To create a new MQSeries queue, run the MQSC command on the MQSeries computer, ie **RUNMQSC** followed by *IBMNT* followed by **DEFINE QLOCAL(IBM.NT.QUEUE)**.

To create a new MSMQ queue on Windows 2000, create a foreign queue in Win2K, as outlined earlier. Remember to use *IBMNT* for your MQSeries Queue Manager name and *IBM.NT.QUEUE* for the MQSeries queue name.

To test the MQSeries Client definitions

- Start the *MQSRRECV* program: open the command prompt and go to the Bridge samples directory, which, by default, is in *C:\Program Files\Host Integration Server\System*.
- Start the program by typing the command **MQSRRECV IBMNT IBM.NT.QUEUE**.
- If the program starts successfully it displays the message 'Use **<CTRL-C>** to stop!'. If you do not receive this message then you can view the channel file in Windows Notepad and, if the file does not exist or it contains no entries, delete the *AMQCLCHL.TAB* file from the MQSeries server and execute the channel file steps again. If the program still fails to start, try overriding the channel file settings by issuing the following command: **SET MQSERVER=<channel name>/<transport type>/<connection name>[(<port number>)]**. For example, in TCP/IP use: **SET MQSERVER=IBMNT_CN/TCP/10.10.10.5(1414)**. With SNA, use **SET MQSERVER=IBMNT_CN/LU62/MQSCPIC. Press CTRL+C** to stop the *MQSRRECV* program.

To start the MSMQ-MQSeries Bridge

- Open the Bridge Manager, right-click on the Bridge Service icon, and click Start.
- In the Manager display, all four message pipes should start showing a green arrow.
- If the message pipes don't start, check the event log.

The MQSeries-MSMQ message pipes may fail if another Bridge is

© 2001. Reproduction prohibited. Please inform Xephon of any infringement.

currently using the same transmission queue, or the Bridge Service is terminated abnormally, causing the MQSeries server to behave as if the Bridge is still using the corresponding XMIT queue. You must restart the MQSeries Queue Manager or configure the Keep Alive option in your MQSeries server.

To send test messages from MSMQ to MQSeries

- On the Windows computer, open two MS-DOS windows. Change both windows to the server installation directory, which, by default, is *C:\Program Files\Host Integration Server\System*.
- Start the MQSeries receiver program in the first window with the command **MQSRRECV IBMNT IBM.NT.QUEUE**.
- In the second window, send ten messages from MSMQ to MQSeries with the command **MSMQSEND IBMNT\IBM.NT.QUEUE**.

If the test succeeds, the first window displays the ten messages that it receives. Stop the *MQSRRECV* program by pressing **CTRL + C** in the first window.

To send test messages from MQSeries to MSMQ

- Open two MS-DOS windows. Go to the default installation directory, which, by default, is *C:\Program Files\Host Integration Server\System*.
- In the first window, start the MSMQ receiver program. Send from MQSeries with the command **MQSRSend IBMNT MSBRIDGE1 MSBRIDGE1.QUEUE**.
- Receive in MSMQ with the command **MSMQRecv MSBRIDGE1\MSBRIDGE1.QUEUE.**

Figure 1 shows a typical configuration, taken from Microsoft's own documentation, and Table 1 lists the settings for a typical configuration. The names in Figure 1 (*MSBRIDGE1*, *IBMNT_CN*, *IBMNT*, etc) are arbitrary, but you must use the same name for the same entity at different locations in the configuration. For example, if the MSMQ-MQSeries Bridge computer is called *MSBRIDGE1*, you must reference this same name in the MSMQ-MQSeries Bridge Manager and in MQSeries.

© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

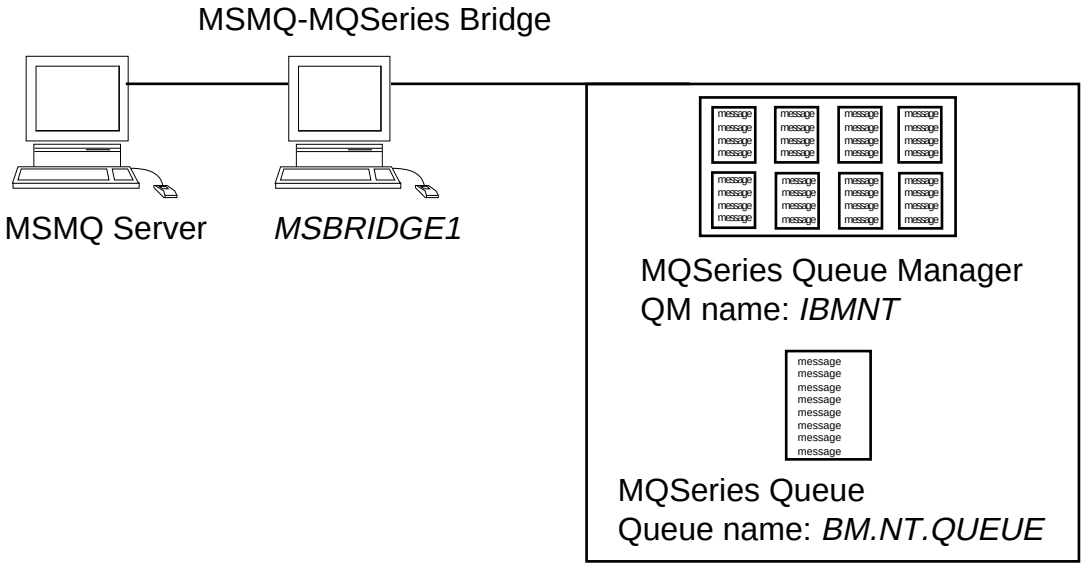

Figure 1: Overview of a typical configuration

Define in MSMQ	Foreign connected network or foreign site	*IBMNT_CN*
	Foreign computer	*IBMNT*
	Foreign queue	*IBM.NT.QUEUE*
Define in MSMQ-MQSeries Bridge		
MSMQ-MQSeries Bridge properties	MQI channel	*IBMNT_CN*
	Transport type	*TCP/IP or LU 6.2*
Foreign Connected Network or Foreign Site properties	CN name	*IBMNT_CN*
	MQSeries QM	*IBMNT*
	Reply to QM	*MSBRIDGE1*
Message pipe properties	MQSeries MSMQ	Normal
	MQSeries MSMQ	High
Transmission queue	*IBM.NT_CN.XMITQ*	*IBMNT_CN.XMITQ.HIGH*
Import or define in MQSeries	Transactional service	Non-transactional service
Transmission queue	*IBMNT_CN.XMITQ*	*IBMNT_CN.XMITQ.HIGH*
Queue manager alias	*MSBRIDGE1*	*MSBRIDGE1%*
	Model queue	*Q2Q_SYNC_Q*
	MQI channel	*MQS_CN*
	Transport type	TCP/IP or LU 6.2

Table 1: Settings for a typical configuration

This series of articles will conclude next month with an examination of Bridge administration and performance issues.

Stephen Ibaraki, ISP
Chief Architect, iGen Knowledge Solutions (Canada) © Xephon

© 2001. Reproduction prohibited. Please inform Xephon of any infringement.

MQ news

IBM has announced V2R4 of its Tivoli Manager for MQSeries. With a policy-based management approach, the product integrates IBM's MQSeries messaging software into broader systems, linking management tasks to databases, applications, and other middleware that spans host and distributed environments.

It can manage a life-cycle from a central location across geographically dispersed systems, and it can install, configure, and administer multiple MQSeries objects from a central location.

Among its key features are support for MQSeries V5.2 for OS/390, OS/400, and Windows 2000. It offers improved usability through its GUI, with easily defined customization and default values displayed where appropriate.

It supports automated dead letter queue handling on all supported platforms except OS/400, and provides Cloning Queue Managers to endpoints for faster, easier configuration on distributed platforms only, and OS/390 statistical data on the internal performance of queue managers.

For further information contact your local IBM representative, or:
Tivoli Systems, Sefton Park, Bells Hill, Stoke Poges, Bucks, SL2 4JS, UK.
Tel: +44 1753-780-000
Fax: +44 1753-780-899
Web: http://www.tivoli.com

* * *

Progress Software has announced that its SonicMQ messaging server is soon to be delivered with a wireless messaging infrastructure. Called SonicAir, it will support transitions from wired to wireless networks, across multiple wireless providers.

It supports multiple wireless protocols including 802.11 B, Cellular Digital Packet Data (CDPD), Motient (DataTac), and Bell South Wireless Data (Mobitex). It dynamically selects the best available protocol for message delivery.

Through its Dynamic Routing Architecture (DRA), the product allows distributed applications to be developed and deployed without regard for connection type or network topology. It makes use of current and emerging connectionless communication services and networks.

For further information contact:
Progress Software, 14 Oak Park, Bedford MA 01730, USA
Tel: +1 781 280 4000
Fax: +1 781 280 4095
Web: http://www.progress.com

Progress Software, The Square, Basingview, Basingstoke, Hants RG21 2EQ, UK
Tel: +44 1256 816668
Fax: +44 1256 463226

* * *