mqsnoop v2.0.0

Tony Madden
tm@mqware.com

1. Overview

In multi-queue manager environments it can be time consuming resolving MQ objects across queue managers and determining the location of problems.

mqsnoop helps reduce the effort by taking a queue name and dynamically resolving it to its ultimate destination or destinations. The state of all intermediate objects is also displayed.

For example:

> Suppose a remote queue APPQ1 on queue manager QM1 resolves to alias queue APPQ2 on queue manager (transmit queue) QM2 which resolves to a triggered queue APPQ3.
>
> Running mqsnoop against APPQ1 will display information about APPQ1; the transmit queue QM2; the sender channel which serves the transmit queue; the alias queue APPQ2 on queue manager QM2; and the local queue APPQ3, along with information about the associated initiation queue and process definition..

mqsnoop proceeds by interrogating queue objects. If a queue object is a remote definition, or a clustered definition, then mqsnoop will resolve, and determine the location of, the remote queue manager (or managers). mqsnoop will then client connect to the remote queue manager (or managers) and continue resolution. Resolution ends when a local queue is found. In the case of multiple ultimate destinations then multiple resolutions take place, each ending when a local queue is found.

mqsnoop can access OS390 queue managers, both those with the Client Attach feature and those without.

mqsnoop will also investigate clusters, displaying constituent queue managers and relevant status.

With minimal setup, mqsnoop can check communications between all queue managers in a cluster. It will check that every cluster queue manager can successfully send a message to every other queue manager in the cluster via the cluster network.

2. Environment

mqsnoop requires MQSeries 5.2 client software.

Executables were complied under Windows NT 4 and Sun Solaris 5.7.

3. Arguments

mqsnoop uses switch style arguments.

Some arguments can be specified multiply. These are indicated in the text with a double asterisk. All other arguments can only be specified once – if they are specified more than once then only the first instance is used.

Arguments can be specified in any order.

3.1 Command Line Arguments

3.1.1 Required Arguments

-c *ClusterName*

The cluster name which mqsnoop will resolve.

-q *QueueName*

The queue name which mqsnoop will resolve.

3.1.2 Optional Arguments

-cd *ChannelName/TrpType/ConnectionName/QmgrKey* **

Specifies a connection definition.

ChannelName	SvrConn channel name. Optional parameter. Defaults to the default channel name
TrpType	Transport Type. Optional parameter. Defaults to TCP)
ConnectionName	Connection Name
QmgrKey	The name of the queue manager targeted by the connection definition.

The four constituent parameters are delimited by "/". Note that trailing delimiters can be optionally omitted.

See the section "**Connection Process**" for more details of the processing of connection definitions.

-e *OutputFormat*

The format of the displayed data. The options are:

v display information in verbose format (default)
f display final information only.

See the section "**Output Formats**" for more details.

-l *LogFile*

The fully qualified name of a log file. Output data will be written to the logfile as well as to stdout. To avoid writing to standard out, pipe the stdout to a file.

-m *InitialQueueManagerName*

If the initial mqsnoop client connect is using a channel table, then the *InitialQueueManagerName* argument can be used to specify a queue manager name which will be used as an index into the channel table.

If mqsnoop is using connection definitions (-cd arguments) then the *InitialQueueManagerName* argument is matched against the *QMgrKey* parameter of a **–cd** connection definition argument.

InitialQueueManagerName is ignored if an MQSERVER environment variable is being used for the initial client connection.

-p *ParmFile*

A fully qualified parameter file name. All mqsnoop arguments (except –p) can be specified in the parameter file. Unchanging arguments (for example, OS390 queue manager information) can be specified in a parameter file while volatile arguments (like queue name) can be specified on the command line.

-r *ReplyToQueuePrefix*

This argument is used when accessing OS390 queue managers which do not have the Client Attach feature. See "**Accessing OS390 Queue managers**" section.

- *DefaultSvrConnChannelName*

The name of the default SvrConn channel name. If this argument is not specified then the default SvrConn channel name is SYSTEM.DEF.SVRCONN.

-u *DefaultContextUserId*

This argument is used in conjunction with proxy access to OS390 queue managers. See the "**Accessing OS390 Queue managers**" section.

-x *OS390_ProxyConnectionInformation* **

This argument is used when accessing OS390 queue managers which do not have the Client Attach feature. See the "**Accessing OS390 Queue managers**" section.

-z

Include SvrConn channel names when displaying connection information. By default, SvrConn channel names are not displayed.

3.1.3 Optional Arguments When Performing Cluster Communication Tests:

See the section "**Cluster Communication Test**" for more details.

-qc

mqsnoop will check all queue managers to check that all the required cluster test queues are defined. If any are not defined then mqsnoop will define them.

-qd

mqsnoop will check all queue managers to for cluster test queues are defined. If any are defined then mqsnoop will delete them. Note that the Purge option will be used for the delete.

-t

Specifies that a cluster communication test is to be performed

-tc

Specifies that a cluster communication test is to be performed. Prior to the communication test, mqsnoop will check all queue managers to check that all the required cluster test queues are defined. If any are not defined then mqsnoop will define them.

-v *Prefix*

The prefix of the cluster test queue. If there is no associated value then the cluster test queue has no prefix.

-w *WaitInterval*

Waitinterval in milliseconds for cluster testing. If a wait interval is required (due to slow network or time taken for cluster channels to start) then this argument is used to specify it. Default value is zero seconds (but see "**Cluster Communication Test**" section for details of mqsnoop sleep intervals).

3.2 Environment Variable MQSNOOP_ARGUMENTS

An environment variable MQSNOOP_ARGUMENTS can be set/exported. All mqsnoop arguments can be specified in the variable value.

3.3 The order in which arguments are processed.

In the event of duplication of arguments, precedence is in the following order (highest first):

1. Command line arguments
2. Arguments contained within the MQSNOOP_ARGUMENTS environment variable
3. Parameter file arguments

If mqsnoop is commonly used from the same machine then it is recommended to put arguments which are static (for example, connection definitions) in a parameter file. This parameter file can then be specified with a –p argument. The mqsnoop call itself is thus made simpler.

For even more simplicity, the parameter file argument itself can be specified in the MQSNOOP_ARGUMENTS environment variable.

.

4. Connection Process

4.1 The Logic Flow of the Initial Client Connection

It is useful to understand how mqsnoop decides which queue manager to connect to initially. The arguments which are relevant to the initial client connection are:

-cd *ChannelName/TrpType/ConnectionName/QMgrKey*

-m *InitialQueueManagerName*

-s *DefaultSvrConnChannelName*

The following logic steps are taken:

- If one or more connection definition arguments (**-cd**) are supplied:

  If an *InitialQueueManagerName* argument (**-m**) is supplied then the connection definition arguments are searched. If a connection definition is found with a *QmgrKey* parameter value equal to the *InitialQueueManagerName* argument, then that connection definition is used for the initial connect. If no connection definition match is found, then an error is returned.

  If an *InitialQueueManagerName* argument (**-m**) is not supplied, then the first connection definition argument is used for the initial connect. Note that command line arguments are processed in preference to parameter file arguments so a connection definition defined on the command line will (in the absence of an *InitialQueueManagerName* argument) override any connection definitions in a parameter file.

  If the chosen connection definition does not specify a SvrConn channel name, then the default SvrConn channel name is used. This is the value of the *DefaultSvrConnChannelName* argument (if specified) or "SYSTEM.DEF.SVRCONN".

- If no connection definition arguments (**-cd**) are supplied:

  If an MQSERVER environment variable has been defined then this is used for the initial client connect.

  If the MQSERVER environment variable is not defined then the MQ channel table will be used (assuming the local environment has been set up with a channel table and associated environment variables).
  If an *InitialQueueManagerName* argument (**-m**) is supplied then this will be used in the MQ connection call to index into the channel table.

4.2 Specifying Server Connection Channels for Client Connections

mqsnoop users need the ability to specify SvrConn channel names for any or all queue managers which might be involved in an mqsnoop queue or cluster resolution. The connection definition arguments (**-cd**) serve this purpose.

mqsnoop will attempt to client connect to a queue manager using the appropriate connection name. Prior to connection, mqsnoop will interrogate the supplied connection definition arguments to find one with a *ConnectionName* property which matches the connection name of the target queue manager. If a connection definition is found and if the *ChannelName* property of the chosen connection definition is non-blank, then it is used as the channel name for the client connection. If the value is blank (or if no connection definition is found), then the default SvrConn channel name value is used instead.

5. Setting up the mqsnoop environment

If mqsnoop is to be invoked from a static location, then it is worth performing some basic set-up which will make usage of mqsnoop very simple. The steps are:

1. **Define and load a parameter file**

   Create a parameter file and put all static definitions in it.

   Static definitions will include connection definitions for all the queue managers which are likely to be involved in mqsnoop queue or cluster resolution. For each connection definition, specify a *QMgrKey* value equal to the associated queue manager name.

   Other static definitions, like default SvrConn channel name, should also be put in the parameter file.

2. **Create an MQSNOOP_ARGUMENTS environment variable**

   The MQSNOOP_ARGUMENTS will contain the parameter file argument, which gives the fully qualified name of the parameter file.

Having performed those two steps, mqsnoop can be invoked with only a few arguments.

5.1 An Example of mqsnoop Set-up:

A Middleware environment consists of four queue managers, QM00001, QM00002 and QM00003. The mqsnoop set-up is as follows:

- The parameter file for mqsnoop is /home/mqutil/mqsnoop.parm
- The contents of /home/mqutil/mqsnoop.parm are:

  -cd //150.43.4.21/QM00001
  -cd //150.43.4.22/QM00002
  -cd ADM.SVRCONN.CHL//150.43.4.56/QM00003

  Note 1: transport type property for all connection definitions is null, so defaults to TCP.
  Note 2: SvrConn channel for QM0001 and QM0002 is null, so connection to both queue managers will use SYSTEM.DEF.SVRCONN as the channel name.

- An MQSNOOP_ARGUMENT environment variable has been set up with the value:

  –p /home/mqutil/mqsnoop.parm

With this set-up, if an mqsnoop is required on queue QTEST1 on queue manager QM0002, then the mqsnoop call would be:

mqsnoop –q QTEST1 –m QM0002

5.2 Authorisation

mqsnoop client connects to queue managers. Briefly, the mqsnoop user, or the MCAUSER on the appropriate SVRCONN channel, must have the appropriate authority to inquire on the relevant MQSeries objects.

See the MQSeries documentation for full details on client security.

6. Accessing OS390 Queue Managers

mqsnoop interrogates queue managers for information about queue manager objects.

On distributed platforms, the MQ Administrative Interface (MQAI) is used. OS390 queue managers differ from distributed queue managers in that the MQAI can not be used. Instead, MQSC format administrative request messages are written to a queue called "SYSTEM.COMMAND.INPUT". The associated response messages contain information which looks very much like the output from the runmqsc utility.

6.1 Set-up Requirements on OS390

The OS390 queue manager must have a command server running.

6.2 Accessing OS390 Queue Managers which have the Client Attach feature

No special action is required for mqsnoop to client connect to OS390 queue managers which have the Client Attach feature.

6.3 Accessing OS390 Queue Managers which do not have the Client Attach feature

It is not possible to client connect to an OS390 queue manager which does not have the Client Attach feature. The solution is to use a proxy queue manager which has connectivity (via channel pairs) with the target OS390 queue manager in question. This way, administrative requests can be sent to the SYSTEM.COMMAND.INPUT queue on the target OS390 queue manager via the proxy queue manager.

Administrative requests are sent to the target OS390 queue manager using the "Queue at QMgr" method, where QMgr is a transmit queue name.

Proxy access can also be used for accessing OS390 queue managers which do have the Client Attach Feature but which are not accessible to mqsnoop users via SvrConn channels .

6.3.1 Specifying Proxy Information in the –x argument

The mqsnoop argument used to specify proxy information is **–x**. The argument value can contain from one to four parameters, delimited by '/'. Normally, only the first two parameters are necessary:

-x *OS390_ConnectionName/ ContextUserId / ProxyConnectionName/ OverrideTransmitQueue*

Parameters must be separated by '/' delimiters. Trailing delimiters can optionally be omitted.

One **–x** argument should be specified for each OS390 queue manager which requires proxy access.

The meaning of each parameter is described next.

OS390_ConnectionName

The connection name for the target OS390 queue manager

ContextUserId

The target OS390 queue manager command server will receive administrative requests under a userid. This userid will be either the userid of the mqsnoop user, or the value of the MCAUSER property of the SvrConn channel used for the current mqsnoop client connection to the proxy queue manager.

This userid will be used as the identity context by the OS390 command server when writing responses to the reply to queue (i.e. the transmit queue associated with the proxy queue manager). If the userid does not have authority to open or put to the reply to queue, then the response will not be received by mqsnoop.

This problem, where it exists, can be avoided by specifying a *ContextUserId* value. If this is done then mqsnoop will write the request message to the proxy command input queue under the identity context of the *ContextUserId* value. Subsequent replies will be written by the command server under the same id.

If the *ContextUserId* component is null then mqsnoop will use the Default Context UserId value for context identity (see the "**Default Context UserId**" section).

If the identity context is required to be null, regardless of the value of the Default Context UserId, then set *ContextUserId* to two single quotes.

ProxyConnectionName

This parameter is normally set to null.

The only time *ProxyConnectionName* is necessary is when the initial MQ object which mqsnoop is resolving resides on an OS390 queue manager which does not have the Client Attach feature. In this situation, the very first connection which mqsnoop needs to make is to a proxy queue manager. The connection name for this proxy queue manager must be specified in the *ProxyConnectionName* parameter. *ProxyConnectionName* is a required parameter in this situation.

In all other situations, the *ProxyConnectionName* value is ignored.

OverrideTransmitQueue

This parameter is normally set to null.

See " **Communicating with the target OS390 queue manager from the proxy queue manager** " for details on how this parameter value is interpreted.

6.3.2 Default Context UserId

The mqsnoop argument used to specify the Default Context UserId is **–u**.

If the **–u** argument is not set, then the Default Context UserId value is null.

6.3.3 Communicating with the target OS390 queue manager from the proxy queue manager

Administrative requests are sent from the proxy queue manager to the target OS390 queue manager using the "Queue at QMgr" method, where QMgr is a transmit queue name.

The transmit queue name is determined by performing the following steps in sequence. If a step is successful then subsequent steps are not performed

STEP 1:

If the **–x** argument specifies an *OverrideTransmitQueue* value, then the *OverrideTransmitQueue* value is used as the transmit queue name for administrative requests.

OverrideTransmitQueue would only be set if the channel servicing a remote queue definition could not be used for sending administrative requests to the target OS390 queue manager. In this situation,

OverrideTransmitQueue would provide the name of a transmit queue servicing an alternative channel to the target OS390 queue manager.

STEP 2

If mqsnoop needs to access the target OS390 queue manager because it has resolved a remote queue definition, then the RQMNAME property of that remote queue definition is used as the transmit queue for administrative requests

STEP 3

This step is performed if mqsnoop is resolving a cluster queue definition, or if mqsnoop is doing an initial connection to a proxy queue manager. In either case, there is no remote queue definition available, so RMQNAME can't be used as the transmit queue. Instead, all channels on the proxy queue manager are queried until a channel is found with a ConnectionName property equal to the target OS390 queue managers connection name. The transmit queue name property of that channel is used as the transmit queue name for administrative requests.

The target OS390 queue managers connection name is given by the *OS390_ConnectionName* parameter of the **–x** argument.

If multiple channels use the target OS390 queue manager connection name then the transmit queue of one of those channels will be chosen.

6.3.4 Specifying ReplyToQueue Prefix for Proxy Access

Administrative replies from an OS390 command server are received on a temporary dynamic queue.

The **–r** argument can be used to specify a prefix for the dynamic queue name.

By default, the name is entirely determined by MQSeries (that is, the dynamic queue name value on the MQ queue open call is "*").

The model queue used for defining the dynamic queue is SYSTEM.DEFAULT.MODEL.QUEUE.

6.3.5 Authorisation and Proxy Access

If a context userid is specified then the mqsnoop user, or the MCAUSER for the relevant SvrConn channel, must have the authority to set Identity Context when writing to the proxy command input queue.

Authority to create temporary dynamic queues is also required.

6.3.6 OS390 Proxy Example 1

- A remote queue called BUS01, on queue manager PSYS01, has an RQMNAME value of QM01.
- A channel exists on PSYS01 with a transmit queue of QM01. The channel has a connection name of mvsa(1313).
- Mvsa(1313) is the connection name for queue manager QM01,an OS390 queue manager which does not have the Client Attach feature.
- The proxy queue manager for QM01 will be PSYS001 (which has connectivity with QM01 via transmit queue QM01).
- The context userid is TU47.
- The temporary dynamic reply to queue (on PSYS0001) which is to be used for reply messages is required to have the prefix "TMP01"

The relevant mqsnoop arguments are:

-x mvsa(1313)/ TU47 -r TMP01

Note that:

1. The *ProxyConnectionName* property of the –x argument is null. This means that mqsnoop will use the currently connected queue manager (i.e. the one with a definition for BUS01) as the OS390 proxy queue manager.
2. The *OverrideTransmitQueue* property of the –x argument is null. This means that the transmit queue used for communicating with QM01 will be the RQMNAME value of the remote queue BUS01.
3. Trailing delimiters on the **–x** argument are omitted. This is optional.

6.3.7 OS390 Proxy Example 2

- The Middleware environment for an mqsnoop includes two OS390 queue managers, QM01 and QM02, neither of which have the Client Attach feature.
- QM01 has connection name 13.4.4.1
- QM02 has connection name 13.4.55.3
- The context userid which will be used for queue managers is PR0004

The relevant mqsnoop arguments are:

-x 13.4.4.1 -x 13.4.55.3 -u PR0004

Note that:

1. The **–u** argument (Default Context User Id) specifies the PR0004 userid. mqsnoop will use the default Context UserId for all OS390 Proxy access.
2. Trailing delimiters on the **–x** arguments are omitted. This is optional.

7. Cluster Communication Test

mqsnoop can check that every queue manager in a cluster can send messages to local definitions of cluster queues on every other queue manager.

Note - the Cluster Communication Test will not interrogate OS390 queue managers which do not have the Client Attach feature.

7.1 How the Cluster Communication Test works

Assume a cluster called CLUS1 which consists of two queue managers, QM1 and QM2. The mqsnoop cluster test setup procedure has been followed, so the following queues exist and are members of CLUS1:

MQSNP_CLUS1_QM1 on queue manager QM1
MQSNP_CLUS1_QM2 on queue manager QM2

To call mqsnoop to test this cluster, the following command is issued (assuming an MQSERVER variable or channel table exist and that the initial connect is to a member of the CLUS1 cluster):

mqsnoop –c CLUS1 –t

mqsnoop will then perform the following steps:

1. First, query the cluster CLUS1. Store information about the constituent queue managers (QM1 and QM2).
2. Connect to each queue manager in the cluster (excluding the currently connected queue manager), construct the mqsnoop test queue name for that queue manager and put a text message to it. The text message must consist of the name of the currently connected queue manager.
3. Sleep for the waitinterval plus 5 seconds to give messages time to propagate across the cluster.
4. Reconnect to each queue manager in turn and read the local mqsnoop test queue and check that there is a message from every other queue manager in the cluster.
5. Report the findings.

All messages are put and got using the same message id, which is a value unique to the current mqsnoop operation.

7.2 Administration for Cluster Communication Testing

7.2.1 Cluster Communication Test Queue Naming Standard

An mqsnoop communication test queue must be resident on each queue manager of the cluster. The queue name should be of the format [p*refix*]_[*clusterName*]_[*queueManagerName*], where:

prefix	"MQSNP" by default. Can be overridden using the –v argument. If the –v argument does not have an associated value then *prefix* is null and the first '_' is not included in the queue name
clusterName	The name of the cluster
queueManagerName	The name of the queue manager on which the queue is defined

For example:

> "MQSNP_TC1_QMA21" is a queue name defined on queue manager QMA21 and a member of cluster TC1. This queue name has the default prefix, so calls to mqsnoop do not need to specify a –v argument.
>
> "ACP1_ADMIN4_QMA21" is a queue name defined on queue manager QMA21 and a member of cluster ADMIN4. This queue name has a non-default prefix, so calls to mqsnoop need to include the argument/value "-v ACP1".
>
> "TC2_QM3" is a queue name defined on queue manager QM3 and a member of cluster TC2. This queue name has no prefix, so calls to mqsnoop must include the –v argument with no associated value.

The resultant queue, once defined, must be added to the relevant cluster.

NOTE: the length of the mqsnoop test queue name can end up greater than 48 bytes. If this is the case then cluster communication testing cannot be performed.

7.2.2 Automatic Setup of Queues using the –qc argument

If mqsnoop is run with the option –qc then all cluster queue managers will be interrogated for the presence of the requisite mqsnoop communication test queue. If the queue is not present then mqsnoop will create it and add it to the cluster.

7.2.3 Automatic Setup of Queues using the –tc argument

If mqsnoop is run with the option –tc then all cluster queue managers will be interrogated for the presence of the requisite mqsnoop communication test queue. If the queue is not present then mqsnoop will create it and add it to the cluster.

If any queues were defined, then mqsnoop will sleep for the waitinterval plus 5 seconds to give the new queue definitions time to propagate across the cluster.

mqsnoop will then run the communication test.

It is possible that the queue definitions will not have propagated through the cluster and hence the communication test may fail with MQRC_OBJECT_NOT_FOUND errors. To help prevent this, mqsnoop will retry communication tests which fail with MQRC_OBJECT_NOT_FOUND errors. A maximum of 5 attempts will be made at intervals of two seconds. Retry attempts will only occur if the current instance of mqsnoop has defined one or more test queues.

7.2.4 Manual Setup of Queues

The mqsnoop communication test queues can be set up manually as follows:

7.2.5 Removing Cluster Communication Test Queues

The mqsnoop test queues can be left in place, once defined, or they can be deleted.

To delete the queues, call mqsnoop with the **–qd** argument.

The **–v** argument can be used to specify the prefix for the queues to be deleted.

Note that the clean up uses the Purge option, so queues with messages on them are deleted.

Note that the mqsnoop test queues will have an open output count of greater than zero for a period of time after running a communication test. The queues can not be deleted until the open output count is zero.

7.2.6 Authorization.

For cluster communication testing, the mqsnoop user, or the MCAUSER on the SvrConn channel, must have authority to read from and write to mqsnoop communication test queues.

If automatic creation of communication test queues is required, then the mqsnoop user, or the MCAUSER on the SVRCONN channel, must have authority to create mqsnoop communication test queues.

If deletion of communication test queues is required, then the mqsnoop user, or the MCAUSER on the SVRCONN channel, must have authority to delete mqsnoop communication test queues.

8. Error Messages and Return Codes

8.1 Return Codes

The mqsnoop executable ends with return code zero when successful.

If mqsnoop has errors (this includes being unable to resolve an object) then it ends with return code 12.

8.2 Error Messages

Errors are displayed with the value “ERROR” in columns 1 to 5. Error messages (apart from OS390 Access error messages) are self-explanatory.

8.3 OS390 Access Error Messages

OS390 Access error messages begin with “ERROR: OS390 Access Error *nnnn*” where *nnnn* is an error code. The error code values are:

4001 Unable to open the command input queue. The text message contains the MQ error.

4002 Unable to open (create) the temporary dynamic reply to queue which will be used to receive responses from the OS390 command server. The text error message contains the MQ error.

4003 No responses from the OS390 command server have been received on the reply to queue. The text message contains the MQ error (2033). Causes of the error could include any of the following:

- For proxy access, the request message could not be transmitted to the SYSTEM.COMMAND.INPUT queue on OS390. This is probably due to communication problems between the proxy and the target OS390 queue managers.
- The command server is not running on OS390
- The response from command server could not be returned. This is probably due to either communication problems between the proxy and the target OS390 queue managers; or authorisation problems when the OS390 command server tries putting responses to the relevant transmit queue

4004 Unable to put command messages to the command input queue. The text message contains the MQ error .

9. Output Formats

9.1 Resolving Queues

mqsnoop will resolve a given queue name. Information about intermediate queue managers, channels and relevant queue objects are displayed, along with the ultimate local queue objects.

The displayed output can be verbose or final. Verbose output displays all the intermediate information. Final output only displays ultimate local queue objects and associated queue manager information.

9.1.1 Verbose Output

This output shows an mqsnoop against a remote queue TEST2 which resolves, via a queue alias, to a local queue called TEST3.

The initial connect is to a queue manager QMA21_1 via connection name tp_a21m(1421) and default transport type (TCP) and SvrConn channel (SYSTEM.DEF.SVRCONN).

Note that the remote queue TEST2 is a member of cluster TC1. TEST2 is defined locally on QMA21_1, so mqsnoop has resolved the local definition. If TEST2 had not been defined locally, then mqsnoop would have resolved the cluster queue into one or more queue objects on one or more remote queue managers.

This snoop highlights the following potential problems:

- The channel QMA21_1.QMA21 is stopped.
- The transmit queue QMA21 has a depth of 2.
- The triggered local queue TEST3 uses an initiation queue called INITQ_1 which has an open input count of zero, so no trigger monitor is running against it.

```
E:\dev\sniff\Release>mqsnoop -q TEST2 -cd //tp_a21m(1421)

mqsnoop v2.0.0.    Resolve Queue TEST2

Snooping ......

************************************************************
*************** QUEUE MANAGER CONNECTION **********************

Name: QMA21_1 (AIX)
Type: Queue Manager

      Connection Name......... tp_a21m(1421)
------------------------------------------------------------
Name: TEST2
Type: Remote Queue   (Clustered)

      Remote Queue Name........ TEST2
      Remote Queue Mgr Name.... QMA21
      Transmission Queue Name..
      Cluster.................. TC1
------------------------------------------------------------
Name: QMA21
Type: Transmission Queue

      Open Input Count......... 0
      Open Output Count........ 0
      Current Depth............ 2
      Maximum Depth............ 25000
      Triggered................ No
```

```
      Serviced by Sender Channel QMA21_1.QMA21
-----------------------------------------------------------
Name: QMA21_1.QMA21
Type: Sender Channel

      Connection Name......... tp_a21m(1414)
      Transport Type.......... TCP
      Status.................. STOPPED

***********************************************************
*************** QUEUE MANAGER CONNECTION ******************

Name: QMA21 (WINDOWSNT)
Type: Queue Manager

      Connection Name......... tp_a21m(1414)
-----------------------------------------------------------
Name: AQ
Type: Alias Queue

      Base Queue.............. TEST3
-----------------------------------------------------------
Name: TEST3
Type: Local Queue

      Open Input Count........ 0
      Open Output Count....... 0
      Current Depth........... 0
      Maximum Depth........... 5000
      Triggered............... Yes
      Trigger type ........... First
      Trigger Msg Priority..... 0
      Trigger data ...........
      Initiation queue ....... INITQ_1
         InitQ Open Input Cnt.. 0
         InitQ Open Output Cnt. 0
         InitQ Current Depth... 0
         InitQ Maximum Depth... 5000
      Process................. PROCESS_3
         Application Id........ e:\temp\util.exe
         Application Type...... WINDOWSNT
         User Data............. abcdef
         Environment Data...... ex


mqsnoop v2.0.0 ended successfully
```

9.1.2 Final Output

This output shows an mqsnoop against a remote queue TEST2 which resolves, via a queue alias, to a local queue called TEST3. The display format is Final, so only information about the ultimate local queue, and its associated queue manager, is displayed. Compare this output with the Verbose output for the same queue in section 8.1.1.

```
E:\dev\sniff\Release>mqsnoop -q RTEST2 -cd //tp_a21m(1421) -e f

mqsnoop v2.0.0.    Resolve Queue RTEST2    ( Display Final Objects Only )

Snooping ......

***********************************************************
*************** QUEUE MANAGER CONNECTION ******************

Name: QMA21
Type: Queue Manager (WINDOWSNT)

      Connection Name......... tp_a21m(1414)
-----------------------------------------------------------
Name: TEST3
```

```
Type: Local Queue

      Open Input Count......... 0
      Open Output Count........ 0
      Current Depth............ 0
      Maximum Depth............ 5000
      Triggered................ Yes
      Trigger type ............ First
      Trigger Msg Priority..... 0
      Trigger data ............
      Initiation queue ........ INITQ_1
         InitQ Open Input Cnt.. 0
         InitQ Open Output Cnt. 0
         InitQ Current Depth... 0
         InitQ Maximum Depth... 5000
      Process.................. PROCESS_3
         Application Id........ e:\temp\util.exe
         Application Type...... WINDOWSNT
         User Data............. abcdef
         Environment Data...... ex


mqsnoop v2.0.0 ended successfully
```

9.2 Resolving OS390 Queues via a Proxy Queue Manager

This output shows the resolution of a remote queue definition TEST5 on queue manager AQB002. TEST5 resolves to queue TEST2 on the OS390 queue manager AQ03.

AQ03 does not have the Client Attach feature, so the queue manager AQB002 is used as a proxy for AQ03.

AQB002 is accessed via connection name 172.27.4.27(1311), transport type TCP and SvrConn channel SYSX.SVRCONN.

The following information is needed to set up the **–x** OS390 proxy argument:

- AQ03 has a connection name of mvsa(1411).
- The userid under which mqsnoop is running is not recognised on the OS390 instance, so a context userid is required. OS390 userid DEV009, which has the appropriate authorisation, is to be used as the context userid.

Note, in the following output, that the connection information for the OS390 queue manager includes information about the proxy environment.

```
E:\dev\sniff\Release>mqsnoop -q TEST5 -cd SYSX.SVRCONN//172.27.4.27(1311) -x mvsa(1411)/
DEV009

mqsnoop v2.0.0.    Resolve Queue TEST5

Snooping ......

************************************************************
************** QUEUE MANAGER CONNECTION **********************

Name: AQB002 (AIX)
Type: Queue Manager

      Connection Name.......... 172.27.4.27(1311)
------------------------------------------------------------
Name: TEST5
Type: Remote Queue

      Remote Queue Name........ TEST2
      Remote Queue Mgr Name.... AQ03
      Transmission Queue Name..
------------------------------------------------------------
```

```
Name: AQ03
Type: Transmission Queue

      Open Input Count......... 1
      Open Output Count........ 1
      Current Depth............ 0
      Maximum Depth............ 100000
      Triggered................ Yes
      Trigger Type ............ First
      Trigger Msg Priority..... 0
      Trigger data ............ AQB2.AQ03
      Initiation queue ........ SYSTEM.CHANNEL.INITQ
          InitQ Open Input Cnt. 1
          InitQ Open Output Cnt 1
          InitQ Current Depth.. 0
          InitQ Maximum Depth.. 1000
      No Process

      Serviced by Sender Channel AQB2.AQ03
------------------------------------------------------------
Name: AQB2.AQ03
Type: Sender Channel

      Connection Name.......... mvsa(1411)
      Transport Type........... TCP
      Status................... RUNNING

************************************************************
************** QUEUE MANAGER CONNECTION **********************

Name: AQ03
Type: Queue Manager (MVS)

      Connection Name.......... mvsa(1411)

      Proxy Queue Manager...... AQB002 (AIX)
      Proxy Connection Name.... 172.27.4.27(1311)
      Proxy (Context) UserId... DEV009
------------------------------------------------------------
Name: TEST2
Type: Local Queue

      Open Input Count......... 0
      Open Output Count........ 0
      Current Depth............ 0
      Maximum Depth............ 999999999
      Triggered................ No

mqsnoop v2.0.0 ended successfully
```

9.3 Cluster Communication Test

Here is a sample output for a cluster TC consisting of three queue managers – QMA21_1, QMA21_2 and QMA21_3.

This test has specified a cluster test queue prefix of SYS01. The following cluster test queues will therefore have been defined (and added to cluster TC1) before this test was run:

- Queue SYS01_TC1_QMA21_1 defined on queue manager QMA21_1
- Queue SYS01_TC1_QMA21_2 defined on queue manager QMA21_2
- Queue SYS01_TC1_QMA21_3 defined on queue manager QMA21_3

```
E:\dev\sniff\Release>mqsnoop -c TC1 -t –v SYS01 -cd //tp_a21m(1421) -l log.txt
snoop v1.8.22


************************************************************
************** QUEUE MANAGER CONNECTION **********************

Name: QMA21_1
Type: Queue Manager

      Connection Name......... tp_a21m(1421)

************************************************************
Name: TC1
Type: Cluster

      QMgrs in Cluster......... 3

      QMgr..................... QMA21_1
      QMgr..................... QMA21_2
      QMgr..................... QMA21_3
------------------------------------------------------------
Name: QMA21_1
Type: Cluster QMgr Intercommunication Test

      Sending QMgr             Status
      QMA21_2.................. OK
      QMA21_3.................. OK
------------------------------------------------------------
Name: QMA21_2
Type: Cluster QMgr Intercommunication Test

      Sending QMgr             Status
      QMA21_1.................. OK
      QMA21_3.................. OK
------------------------------------------------------------
Name: QMA21_3
Type: Cluster QMgr Intercommunication Test

      Sending QMgr             Status
      QMA21_1.................. OK
      QMA21_2.................. OK

snoop v1.8.22 ended
```