Exploiting the Automatic Client Reconnect feature introduced in WebSphere MQ 7.0.1 – MQI (C Code)

IBM Techdoc:
http://www.ibm.com/support/docview.wss?rs=171&uid=swg27017882

Date last updated: 7-Jun-2010

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

+++ Objective +++

The objective of this techdoc is to show real examples that use MQI (C Code) that exploit the new feature introduced in WebSphere MQ 7.0.1: Automatic Client Reconnect.

There are several mechanisms that specify how a client will do the automatic reconnection. This is the order of reconnection:

1) MQCNO flags for MQCONN – this is not shown in this document (these flags are specified inside an application).

2) If MQSERVER is present, then it is used. It overrides all the options 3 and 4.

3) If the MQSERVER is NOT present, then the next option is to use the Client Configuration File (mqclient.ini).

4) Finally, if none of the above is defined, then the CCDT is used.

+++ Table of Contents +++

+++ References

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp?topic=/com.ibm.mq.csqzaf.doc/cs70190_.htm
MQ V7 Information Center
> Clients
>> Application Programming
>>> Automatic client reconnection

http://www.ibm.com/support/docview.wss?uid=swg27016801
WSTE: WebSphere MQ V7.0 Client Enhancements

++
+++ Chapter 1: Setup of queue managers and clients
++

a) Standalone queue managers

The following standalone queue managers are used:

Host-A: veracruz.y.ibm.com (SuSE Linux Enterprise Server SLES 9), x86 32-bit
Queue manager name: QM_VER Port: 1414

Host-B: aemtux1.z.ibm.com (SUSE LINUX Enterprise Server 9), Power PC 64-bit
Queue manager name: ANGEL_TUX1 Port: 1421

The MQ version in both systems is 7.0.1.1.

b) Multi-instance queue manager

The Automatic Client Reconnect feature is independent of the Multi-Instance Queue Manager feature, both introduced in WebSphere MQ 7.0.1.0.

However, these 2 features are a good complement of each other, and thus, in this document, a multi-instance queue manager is used for some of the example scenarios.
It is NOT necessary for the queue managers to be multi-instance queue managers in order to be used with the automatic reconnect.

In this example, the following multi-instance queue manager will be used:
Queue manager name: QMMI1

The hosts where the instances run are:
Host-1: cbeech.x.ibm.com (SuSE Linux Enterprise Server SLES 10), x86 32-bit
Host-2: veracruz.y.ibm.com (SuSE Linux Enterprise Server SLES 9), x86 32-bit
The listener port in both systems is 1421
The MQ version in both systems is 7.0.1.1.
The user “rivera” is a member of the “mqm” group, thus, it is an MQ administrator.

The active instance is running in Host-1:

```
rivera@cbeech: /var/mqm
$ strmqm -x QMMI1
WebSphere MQ queue manager 'QMMI1' starting.
5 log records accessed on queue manager 'QMMI1' during the log replay phase.
```

```
Log replay for queue manager 'QMMI1' complete.
Transaction manager state recovered for queue manager 'QMMI1'.
WebSphere MQ queue manager 'QMMI1' started.

$ dspmq -m QMMI1 -x
QMNAME(QMMI1)                                    STATUS(Running)
   INSTANCE(cbeech) MODE(Active)
   INSTANCE(veracruz) MODE(Standby)
```

The standby instance is running in Host-2.

```
rivera@veracruz: /home/rivera
$ strmqm -x QMMI1
WebSphere MQ queue manager 'QMMI1' starting.
A standby instance of queue manager 'QMMI1' has been started. The active
instance is running elsewhere.

$ dspmq -m QMMI1 -x
QMNAME(QMMI1)                                    STATUS(Running as standby)
   INSTANCE(cbeech) MODE(Active)
   INSTANCE(veracruz) MODE(Standby)
```

<u>c) Create objects to facilitate the remote access via MQ Explorer and the test scenarios</u>

From the host that has the active instance, create the following objects.
The default port number for the MQ Listener is 1414.

```
$ runmqsc QueueManager

# The following objects are needed to use MQ Explorer remotely
# You may want to customize some of the attributes, such as QMNAME, CONNAME,
port number, etc.

DEFINE LISTENER(TCP.LISTENER) TRPTYPE(TCP) CONTROL(QMGR) PORT(1414)

START LISTENER(TCP.LISTENER)

DEFINE CHANNEL(SYSTEM.ADMIN.SVRCONN) CHLTYPE(SVRCONN)

# The following statements are needed for the test scenarios.

DEFINE QLOCAL(SOURCE)
DEFINE QLOCAL(TARGET)

END
```

d) MQ client and samples

The MQ clients to be tested are:

d.1) MQ 6.0.2.8 from Linux

Samples compiled under MQ 6.0.2.8 from Linux were copied into a host with MQ 7.0.1.1 to test the capability of automatic client reconnection without the need for rebuild.
The tests were successful.

d.2) MQ 7.0.1.1 from Linux

1) Traditional put/get, pub/sub

amqsputc – put message using client mode (not bindings)
amqsgutc – get message using client mode (not bindings)
amqspub – publish
amqssub – subscribe

2) High availability sample programs

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp?topic=/com.ibm.mq.csqzal.doc/fg17235_.htm
WebSphere MQ 7.0.1
> High availability sample programs

"The amqsghac, amqsphac and amqsmhac high availability sample programs use automated client reconnection to demonstrate recovery following the failure of a queue manager.
"

The 3 sample programs are:

PUT:
amqsphac queueName [qMgrName]
 * Puts a sequence of messages to a queue with a 2-second delay between each message.
 * Displays events sent to its event handler.

 * No sync point is used.
 * Reconnection can be made to any queue manager.

GET:
amqsghac queueName [qMgrName]
 * Gets messages from a queue.
 * Displays events sent to its event handler.
 * No sync point is used.
 * Reconnection can be made to any queue manager.

MOVE:
amqsmhac -s sourceQueueName -t targetQueueName [-m qMgrName] [-w waitInterval]
 * Copies messages from one queue to another under sync point, with a default wait interval of 15 minutes after the last message that is received before the program finishes.
 * Sync point is used.
 * Reconnection can be made only to the same queue manager.

<u>e) Several ssh/telnet/command prompt windows are needed</u>

The scenarios are run from a Windows XP machine, which has MQ 7.0.1.1 and uses the tool PuTTy to have remote connections via ssh and/or telnet.

Several ssh/telnet/command prompt windows are needed for the scenarios.
Each window will be numbered in the scenarios to help keep track of the tasks done.

Window-1: host veracruz - start/stop QMgr
Window-2: host veracruz - Put – samples: amqsphac, amqsputc, amqspub
Window-3: host veracruz - Get – samples: amqsghac, amqsgetc, amqssub
Window-4: host veracruz - Move msg from one Q to another - amqsmhac
Window-5: host aemtux1 - start/stop QMgr
Window-6: host cbeech - start/stop QMgr
Window-7: host Windows XP – client samples

f) Shell script in Unix to modify the command prompt to help track the purpose for each window
To help track the proper window while performing the scenario, the command prompt (specified by the PS1 environment variable in Unix) can be modified by a script to indicate the user name, hostname and window number.

You can create this script in the “bin” directory of your userid.

Change the permissions to execute.

```
  chmod 755 set-windowid.ksh
```

Then run the script with a leading dot "." And space to "source-in" the environment variables. For example, the following will be for window number 1:

```
# Window-1 (login as MQ administrator called "rivera" in host "veracruz):
.  set-windowid.ksh  1
```

The result will be a command prompt that looks like this:

```
        rivera@veracruz.Window_1: /home/rivera
```

< begin script >

```
# Usage:  . set-window-id.ksh windowID
# Purpose: To modify the PS1 prompt to indicate:
#  username@hostname.Window_ID current_directory
# For example:
#  rivera@veracruz.Window_1: /home/rivera
#
if [ "$#" -eq 1 ]
then
   export WID=$1
   export SITE=`/bin/hostname -s`
export PS1='$USER@$SITE.Window_$WID: $PWD \$ '
else
   echo "need to provide a single integer"
fi
```

< end script >

++
+++ Chapter 2: Using the “High availability sample programs” (C code) – single queue manager
++

The scenarios in this section show the basic automatic client reconnection capability of MQ V7 clients when working with a single queue manager.
After the clients are connected and exchanging messages, the queue manager is terminated with the flag “-r” to allow the reconnection. Then the queue manager is re-started and the clients automatically reconnect.

Using High availability sample programs

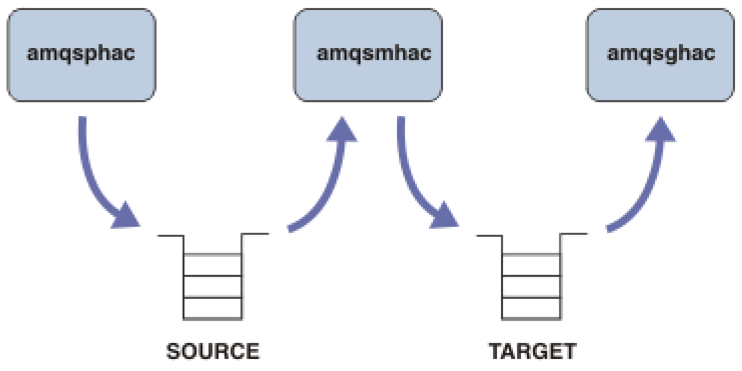
See Chapter 1 for more details on these samples.

PUT: amqsphac queueName [qMgrName]

GET: amqsghac queueName [qMgrName]

MOVE: amqsmhac -s sourceQueueName -t targetQueueName [-m qMgrName] [-w waitInterval]

The following figure shows how these 3 samples work together.
SOURCE and TARGET are the queues that are used.



Customization of the Client Channel Definition Table (CCDT)

In the both hosts run the following runmqsc commands.

This complements / replaces the runmqsc commands mentioned earlier in this document.

```
DEFINE CHANNEL(CHANNEL1) CHLTYPE(SVRCONN)  TRPTYPE(TCP) REPLACE

DEFINE CHANNEL(CHANNEL2) CHLTYPE(SVRCONN)  TRPTYPE(TCP) REPLACE

DEFINE CHANNEL(CHANNEL1) CHLTYPE(CLNTCONN)  TRPTYPE(TCP)  +
     CONNAME('aemtux1.z.ibm.com(1421)') QMNAME(ANGEL_TUX1) REPLACE

DEFINE CHANNEL(CHANNEL2) CHLTYPE(CLNTCONN)  TRPTYPE(TCP)  +
     CONNAME('veracruz.y.ibm.com(1414)') QMNAME(QM_VER) REPLACE

END
```

<u>Scenario 2-A) Testing reconnect to a single standalone queue manager (using CCDT file)</u>

- Open 4 windows in the local host (Host-A): Window-1 thru Window-4
- Open 1 window for the remote host (Host-B): Window-5

++ Stop the local queue manager and start the remote queue manager, to confirm that the samples need to access the remote queue manager.

- In Window-1 veracruz:
Login as an MQ admin:

```
        endmqm -i QM_VER
```

- In Window-5 aemtux1:
Login as an MQ admin:

```
        strmqm ANGEL_TUX1
```

- In Window-1 for the local host (Host-A), verify that AMQCLCHL.TAB client channel definition table (CCDT) file is available in the default location:

```
$ ls -l /var/mqm/qmgrs/QM_VER/@ipcc/AMQCLCHL.TAB
-rw-rw----  1 mqm mqm 2030 2010-01-26 08:13
/var/mqm/qmgrs/QM_VER/@ipcc/AMQCLCHL.TAB
```

- In each of the windows in Host-A, export the environment variable MQCHLLIB to the path to the AMQCLCHL.TAB client channel definition table (CCDT) file:

```
        export MQCHLLIB=/var/mqm/qmgrs/QM_VER/@ipcc/
```

Note:
The default value for the related environment variable is:

```
        export MQCHLTAB=AMQCLCHL.TAB
```

Because we are keeping this name, there is no need to explicitly define this variable.

Ensure to unset these variables, just in case it was defined in the profile:

```
unset MQSERVER
unset MQCLNTCF
```

Note:
To unset a variable in Windows specify a null string to the variable via “=”:

```
set MQSERVER=
```

The following will be explained later on. Ensure to comment out the statements for:

```
ServerConnectionParms
```

… in the Client configuration file:

```
/var/mqm/mqclient.ini
```

++ Start the samples in Host-A

Notice that the queue manager name is NOT specified in the samples.

- In Window-2 (put: amqsphac) issue:

```
$ amqsphac SOURCE ANGEL_TUX1
Sample AMQSPHAC start
target queue is SOURCE
message <Message 1>
message <Message 2>
message <Message 3>
```

- In Window-4 (move: amqsmhac) issue:

```
$ amqsmhac -s SOURCE -t TARGET -m ANGEL_TUX1
Sample AMQSMHA0 start
```

- In Window-3 (get: amqsghac) issue:

```
$ amqsghac TARGET ANGEL_TUX1
Sample AMQSGHAC start
message <Message 1>
message <Message 2>
message <Message 3>
```

- Wait for 15 seconds and observe the text in each window.

- In Window-5 of the remote Host-B, stop the queue manager and allow the clients to reconnect (flag: -r).

Note: In case that you need to expedite the reconnection process, you may need to use the -i flag (immediate). Notice that the AMQ9604 error will be shown for each connected client.

```
$ endmqm -r -i ANGEL_TUX1
Waiting for queue manager 'ANGEL_TUX1' to end.
WebSphere MQ queue manager 'ANGEL_TUX1' ending.
01/21/2010 12:49:40 PM AMQ9604: Channel 'SYSTEM.DEF.SVRCONN' terminated
unexpectedly
01/21/2010 12:49:40 PM AMQ9604: Channel 'SYSTEM.DEF.SVRCONN' terminated
unexpectedly
01/21/2010 12:49:40 PM AMQ9604: Channel 'SYSTEM.DEF.SVRCONN' terminated
unexpectedly
WebSphere MQ queue manager 'ANGEL_TUX1' ended.
```

- Observe the text in each window and verify that each program is trying to reconnect.

- Window-2 (put: amqsphac):

```
message <Message 75>
13:16:47 : EVENT : Connection Reconnecting (Delay: 241ms)
13:16:48 : EVENT : Connection Reconnecting (Delay: 145ms)
13:16:48 : EVENT : Connection Reconnecting (Delay: 2462ms)
13:16:51 : EVENT : Connection Reconnecting (Delay: 4652ms)
13:16:56 : EVENT : Connection Reconnecting (Delay: 8558ms)
13:17:04 : EVENT : Connection Reconnecting (Delay: 18847ms)
```

- Window-4 (move: amqsmhac):

```
13:16:47 : EVENT : Connection Reconnecting (Delay: 58ms)
13:16:47 : EVENT : Connection Reconnecting (Delay: 1032ms)
13:16:48 : EVENT : Connection Reconnecting (Delay: 2020ms)
13:16:50 : EVENT : Connection Reconnecting (Delay: 4601ms)
13:16:55 : EVENT : Connection Reconnecting (Delay: 8256ms)
13:17:03 : EVENT : Connection Reconnecting (Delay: 16628ms)
```

- Window-3 (get: amqsghac):

```
message <Message 75>
13:16:47 : EVENT : Connection Reconnecting (Delay: 245ms)
13:16:47 : EVENT : Connection Reconnecting (Delay: 1300ms)
13:16:49 : EVENT : Connection Reconnecting (Delay: 2439ms)
13:16:51 : EVENT : Connection Reconnecting (Delay: 4306ms)
13:16:55 : EVENT : Connection Reconnecting (Delay: 8920ms)
13:17:04 : EVENT : Connection Reconnecting (Delay: 17981ms)
```

- In Window-1 look at the /var/mqm/errors/AMQERR01.LOG file:

Notice the following errors:

```
01/26/2010 12:30:36 PM - Process(29626.3) User(rivera) Program(amqsghac)
                Host(veracruz)
AMQ9524: Remote queue manager unavailable.
EXPLANATION:
Channel 'CHANNEL1' cannot start because the remote queue manager is not
currently available.
ACTION:
Either start the remote queue manager, or retry the operation later.

01/26/2010 12:30:37 PM - Process(29624.4) User(rivera) Program(amqsphac)
                Host(veracruz)
AMQ9202: Remote host 'aemtux1 (9.42.131.12) (1421)' not available, retry later.
EXPLANATION:
The attempt to allocate a conversation using TCP/IP to host 'aemtux1
(9.42.131.12) (1421)' was not successful.  However the error may be a
transitory one and it may be possible to successfully allocate a TCP/IP
conversation later.
ACTION:
Try the connection again later. If the failure persists, record the error
values and contact your systems administrator. The return code from TCP/IP is
111 (X'6F'). The reason for the failure may be that this host cannot reach the
destination host. It may also be possible that the listening program at host
'aemtux1 (9.42.131.12) (1421)' was not running.  If this is the case, perform
the relevant operations to start the TCP/IP listening program, and try again

01/26/2010 12:30:40 PM - Process(29626.2) User(rivera) Program(amqsghac)
                Host(veracruz)
AMQ9209: Connection to host '9.42.131.12(1421)' closed.
EXPLANATION:
An error occurred receiving data from '9.42.131.12(1421)' over TCP/IP.  The
connection to the remote host has unexpectedly terminated.
ACTION:
Tell the systems administrator.
```

- In Window-5 of the remote Host-B, restart the queue manager:

```
$ strmqm ANGEL_TUX1
WebSphere MQ queue manager 'ANGEL_TUX1' starting.
35 log records accessed on queue manager 'ANGEL_TUX1' during the log replay phase.
```

```
Log replay for queue manager 'ANGEL_TUX1' complete.
Transaction manager state recovered for queue manager 'ANGEL_TUX1'.
WebSphere MQ queue manager 'ANGEL_TUX1' started.
```

- Observe the text in each window and verify that each program reconnected.

Notice that the MQ error log (AMQERR01.LOG) for the queue manager will have some messages:
/var/mqm/qmgrs/ANGEL_TUX1/errors

```
01/26/2010 12:32:43 PM - Process(26813.7) User(rivera) Program(amqrmppa)
                    Host(aemtux1)
AMQ9209: Connection to host 'dyn452040 (9.37.249.176)' closed.
EXPLANATION:
An error occurred receiving data from 'dyn452040 (9.37.249.176)' over TCP/IP.
The connection to the remote host has unexpectedly terminated.
ACTION:
Tell the systems administrator.

01/26/2010 12:32:46 PM - Process(26813.8) User(rivera) Program(amqrmppa)
                    Host(aemtux1)
AMQ9209: Connection to host 'dyn452040 (9.37.249.176)' closed.
EXPLANATION:
An error occurred receiving data from 'dyn452040 (9.37.249.176)' over TCP/IP.
The connection to the remote host has unexpectedly terminated.
ACTION:
Tell the systems administrator.
```

- Window-2 (put: amqsphac):

```
13:17:23 : EVENT : Connection Reconnected
message <Message 76>
13:17:23 : EVENT : Connection Broken
message <Message 77>
message <Message 78>
message <Message 79>
```

- Window-4 (move: amqsmhac):

```
13:17:20 : EVENT : Connection Reconnected
13:17:20 : EVENT : Connection Broken
```

- Window-3 (get: amqsghac):

```
13:17:22 : EVENT : Connection Reconnected
13:17:23 : EVENT : Connection Broken
message <Message 76>
```

- To stop the samples, press <Ctrl-C> in each of the windows.

<u>Scenario 2-B) Testing reconnect to a single standalone queue manager (using MQSERVER)</u>

This is a variation of Scenario 2-A, but using the MQSERVER variable instead of the CCDT file.

++ In each of the windows used in Scenario 2-A, do the following 2 tasks:

- Unset the environment variables that reference the CCDT file:
```
unset MQCHLLIB
unset MQCHLTAB
unset MQCLNTCF
```

- Set the following environment variable to point to a remote queue manager:
```
export MQSERVER=SYSTEM.DEF.SVRCONN/TCP/'aemtux1.z.ibm.com(1421)'
```

<u>Notes on using single quotes with MQSERVER:</u>
1) In Unix, it is necessary to use single quotes around hostname(port) as shown above, or around the entire string:
```
  export MQSERVER='SYSTEM.DEF.SVRCONN/TCP/aemtux1.z.ibm.com(1421)'
```
2) In Windows, do not include the single quotes:
```
  set MQSERVER=SYSTEM.DEF.SVRCONN/TCP/aemtux1.z.ibm.com(1421)
```
3) For more examples of MQSERVER, see “Appendix: Troubleshooting”.

++ Start the samples in Host-A (veracruz).

++ Repeat the same steps as in Scenario 2-A.

Notice that because MQSERVER specifies the access to a particular queue manager, there is no need to add it to the samples:

- In Window-2 (put: amqsphac) issue:
```
$ amqsphac SOURCE
```

- In Window-4 (move: amqsmhac) issue:
```
$ amqsmhac -s SOURCE -t TARGET
```

- In Window-3 (get: amqsghac) issue:
```
$ amqsghac TARGET
```

- Wait for 15 seconds and observe the text in each window.

- In Window-5 for the remote Host-B, stop the queue manager and allow the clients to reconnect (flag: -r).

```
$ endmqm -r -i ANGEL_TUX1
```

- Continue with the rest of the steps of the scenario 2-A:
Re-start the queue manager and observe the reconnection.

Note: The results should be the same as in Scenario 2-A.

- To stop the samples, press <Ctrl-C> in each of the windows.

<u>Scenario 2-C) Testing reconnect to a single standalone queue manager (using client configuration file)</u>

This is a variation of Scenario 2-A and Scenario 2-B. The main difference is neither the CCDT file nor the MQSERVER variable will be used. Instead, the WebSphere MQ client configuration file will be used. This is a feature introduced in MQ V7.

++ Edit the client configuration file:

In the local Host-A (veracruz), modify the client configuration file (the default location is shown below):

```
        /var/mqm/mqclient.ini
```

This file can be also specified by the environment variable: MQCLNTCF

Add the following stanza (which will act as having the MQSERVER environment variable being set) and allowing the reconnectable clients to reconnect.

```
CHANNELS:
  DefRecon=YES
  ServerConnectionParms=SYSTEM.DEF.SVRCONN/TCP/aemtux1.z.ibm.com(1421)
```

Note: Do NOT use single quotes around the connection name! The MQSERVER variable in Unix requires them, but they do NOT work inside the client configuration file.

```
  ServerConnectionParms=SYSTEM.DEF.SVRCONN/TCP/'aemtux1.z.ibm.com(1421)'
```

The above does NOT work and you will get the return code 2538 (MQRC_HOST_NOT_AVAILABLE):

```
        MQCONNX ended with reason code 2538
```

- In each of the MQ client windows used in Scenario A, unset the environment variables that reference the CCDT file and MQSERVER. By not using either of these variables, then the Client Configuration file will be used.

```
unset MQCHLLIB
unset MQCHLTAB
unset MQSERVER
```

++ Start the samples in the local Host-A (veracruz).

- Repeat the same steps as in Scenario 2-B.

- In Window-2 (put: amqsphac) issue:

```
$ amqsphac SOURCE
```

- In Window-4 (move: amqsmhac) issue:

```
$ amqsmhac -s SOURCE -t TARGET
```

- In Window-3 (get: amqsghac) issue:

```
$ amqsghac TARGET
```

- Wait for 15 seconds and observe the text in each window.

- In Window-5 for the Host-B, stop the queue manager and allow the clients to reconnect (flag: -r).

```
$ endmqm -r -i ANGEL_TUX1
```

- Continue with the rest of the steps of the scenario 2-B:
Re-start the queue manager and observe the reconnection.

Note: The results should be the same as in Scenario 2-B.

- To stop the samples, press <Ctrl-C> in each of the windows.

++
+++ Chapter 3: Using the “High availability sample programs” (C code) – 2 separate queue managers
++

The scenarios in this section are similar to the previous chapter, with the exception that 2 queue managers will be used (instead of one).
Two variations will be explored:
- Using 2 standalone queue managers.
- Using a multi-instance queue manager and doing a switchover from the active to the standby instance.

<u>Customization of the Client Channel Definition Table</u>

In the local host (Host-A, “veracruz” in these scenarios), run the following commands, to complement/replace the runmqsc commands mentioned earlier in this document.

- In Window-1 veracruz:

```
$ strmqm QM_VER
```

Note that in CONNAME only a single pair of single quotes should be used to delimit the WHOLE string which includes one or more hostname(port) tokens.

```
        Correct:     CONNAME('ipaddr1(1414),ipaddr2(1414)')
        Incorrect:   CONNAME('ipaddr1(1414) ', 'ipaddr2(1414)')
```

In the incorrect sample, notice that each hostname(port) is surrounded by single quotes.

```
$ runmqsc QM_VER

DEFINE CHANNEL(CHANNEL1) CHLTYPE(CLNTCONN)  TRPTYPE(TCP)  +
CONNAME('aemtux1.z.ibm.com(1421),+
veracruz.y.ibm.com(1414)') QMNAME(ANGEL_TUX1) REPLACE

END
```

Scenario 3-A) Testing reconnect to another standalone queue manager (using MQSERVER) – Failure of sample with sync point

The local Host-A "veracruz" will be used to run the client and the 2nd queue manager. The remote Host-B "aemtux1" will be used to run the first queue manager.

- Open 4 windows in the local host (Host-A): Window-1 thru Window-4
- Open 1 window for the remote host (Host-B): Window-5

++ Stop the local queue manager and start the remote queue manager, to confirm that the samples need to access the remote queue manager (1st queue manager).

- Login to Host-A (veracruz):

```
        endmqm -i QM_VER
```

- Login to Host-B (aemtux1):

```
        strmqm ANGEL_TUX1
```

- In each of the 4 windows in Host-A, set the following environment variable to point to the 2 queue managers (in a single line):

```
export
MQSERVER=SYSTEM.DEF.SVRCONN/TCP/'aemtux1.z.ibm.com(1421),veracruz.y.ibm.co
m(1414)'
```

- Unset the other variables:

```
        unset MQCHLLIB
        unset MQCLNTCF
```

- Comment out the "ServerConnectionParms" entries in the Client configuration file: /var/mqm/mqclient.ini

++ Start the samples in Host-A

- In Window-2 (put: amqsphac) issue:

```
$ amqsphac SOURCE
Sample AMQSPHAC start
target queue is SOURCE
message <Message 1>
message <Message 2>
message <Message 3>
```

- In Window-4 (move: amqsmhac) issue:

```
$ amqsmhac -s SOURCE -t TARGET
Sample AMQSMHA0 start
```

- In Window-3 (get: amqsghac) issue:

```
$ amqsghac TARGET
Sample AMQSGHAC start
message <Message 1>
message <Message 2>
message <Message 3>
```

- Wait for 15 seconds and observe the text in each window.

- In Window-5 of the remote Host-B, stop the queue manager and allow the clients to reconnect (flag: -r).

```
$ endmqm -r -i ANGEL_TUX1
```

- Observe the text in each window and verify that each program is trying to reconnect.

- Window-2 (put: amqsphac):

```
message <Message 75>
13:16:47 : EVENT : Connection Reconnecting (Delay: 241ms)
13:16:48 : EVENT : Connection Reconnecting (Delay: 145ms)
13:16:48 : EVENT : Connection Reconnecting (Delay: 2462ms)
13:16:51 : EVENT : Connection Reconnecting (Delay: 4652ms)
13:16:56 : EVENT : Connection Reconnecting (Delay: 8558ms)
13:17:04 : EVENT : Connection Reconnecting (Delay: 18847ms)
```

- Window-4 (move: amqsmhac):

```
13:16:47 : EVENT : Connection Reconnecting (Delay: 58ms)
13:16:47 : EVENT : Connection Reconnecting (Delay: 1032ms)
13:16:48 : EVENT : Connection Reconnecting (Delay: 2020ms)
13:16:50 : EVENT : Connection Reconnecting (Delay: 4601ms)
13:16:55 : EVENT : Connection Reconnecting (Delay: 8256ms)
13:17:03 : EVENT : Connection Reconnecting (Delay: 16628ms)
```

- Window-3 (get: amqsghac):

```
message <Message 75>
13:16:47 : EVENT : Connection Reconnecting (Delay: 245ms)
```

```
13:16:47 : EVENT : Connection Reconnecting (Delay: 1300ms)
13:16:49 : EVENT : Connection Reconnecting (Delay: 2439ms)
13:16:51 : EVENT : Connection Reconnecting (Delay: 4306ms)
13:16:55 : EVENT : Connection Reconnecting (Delay: 8920ms)
13:17:04 : EVENT : Connection Reconnecting (Delay: 17981ms)
```

- In Window-1 (for Host-A), start the 2nd queue manager:

```
$ strmqm QM_VER
```

- Observe the text in each window and verify that each program reconnected.

The samples in Window-2 (Put) and Window-3 (Get) worked fine and reconnected.

<u>NOTE!!!</u>

However, for Window-4 (amqsmhac – transfer via sync point), after stopping the remote queue manager and starting the local queue manager to force a reconnect, the following return code was issued: 2548
The reason is that this sample uses a sync point and this is not supported during reconnects!

- Window-2 (put: amqsphac):

```
13:17:23 : EVENT : Connection Reconnected
message <Message 76>
13:17:23 : EVENT : Connection Broken
message <Message 77>
message <Message 78>
message <Message 79>
```

- Window-4 (move: amqsmhac):

```
15:40:17 : EVENT : Connection Reconnecting (Delay: 151ms)
15:40:19 : EVENT : Connection Reconnecting (Delay: 0ms)
15:40:19 : EVENT : Connection Reconnecting (Delay: 2165ms)
15:40:21 : EVENT : Connection Reconnecting (Delay: 4747ms)
15:40:26 : EVENT : Reconnection failed
MQGET ended with reason code 2548
MQBACK ended with reason code 2548
MQDISC ended with reason code 2548
Sample AMQSMHA0 end
```

```
$ mqrc 2548
     2548  0x000009f4  MQRC_RECONNECT_FAILED
```

- Window-3 (get: amqsghac):

```
13:17:22 : EVENT : Connection Reconnected
13:17:23 : EVENT : Connection Broken
```

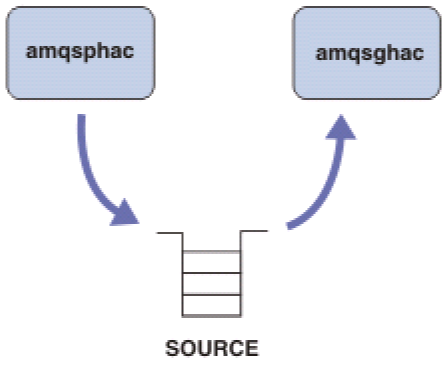
- To stop the samples, press <Ctrl-C> in each of the windows.

<u>Scenario 3-B) VARIATION of Scenario 3-A: Using only the put and get samples (no sync point)</u>

This scenario explores a variation that does not use the sample that employs sync-point.

Because an application (sample amqsmhac) that uses syncpoint cannot be automatically reconnected as shown in Scenario 3-A, the following variation will be explored:

The sample amqsphac will put a message into SOURCE and the sample amqsghac will get the message from SOURCE. In the previous scenario, the get action was on the TARGET queue.



The remote queue manager ANGEL_TUX1 is running.
The local queue manager QM_VER is stopped.

- In Window-2 (put: amqsphac):

```
$ amqsphac SOURCE
Sample AMQSPHAC start
target queue is SOURCE
message <Message 1>
```

>> These messages are being placed into the remote queue manager ANGEL_TUX1 into queue SOURCE:

```
message <Message 2>
message <Message 3>
message <Message 4>
message <Message 5>
```

- Window-4 (move: amqsmhac): not used
The sample uses sync point and it can only reconnect to the same original queue manager.

- In Window-3 (get: amqsghac)
NOTICE that the queue is SOURCE (not TARGET)

```
$ amqsghac SOURCE
Sample AMQSGHAC start
message <Message 1>
```

>> These messages are being read from the queue SOURCE from remote queue manager ANGEL_TUX1

```
message <Message 2>
message <Message 3>
message <Message 4>
message <Message 5>
```

- Wait for 15 seconds and observe the text in each window.

- In Window-5 of the remote Host-B, stop the queue manager and allow the clients to reconnect (flag: -r).

```
$ endmqm -r -i ANGEL_TUX1
```

- Observe the text in each window and verify that each program is trying to reconnect.

- Window-2 (put: amqsphac):

```
message <Message 7>
15:16:54 : EVENT : Connection Reconnecting (Delay: 209ms)
15:16:57 : EVENT : Connection Reconnecting (Delay: 0ms)
15:16:57 : EVENT : Connection Reconnecting (Delay: 2247ms)
15:16:59 : EVENT : Connection Reconnecting (Delay: 4646ms)
15:17:04 : EVENT : Connection Reconnecting (Delay: 8069ms)
```

- Window-3 (get:amqsghac):

```
message <Message 7>
15:16:54 : EVENT : Connection Reconnecting (Delay: 90ms)
15:16:57 : EVENT : Connection Reconnecting (Delay: 0ms)
15:16:57 : EVENT : Connection Reconnecting (Delay: 2348ms)
15:16:59 : EVENT : Connection Reconnecting (Delay: 4671ms)
```

```
15:17:04 : EVENT : Connection Reconnecting (Delay: 9747ms)
```

- In Window-1 of Host-A, restart the 2nd queue manager:

```
$ strmqm QM_VER
```

- Observe the text in each window and verify that each program reconnected.

- Window-2 (put: amqsphac):

```
15:17:04 : EVENT : Connection Reconnecting (Delay: 8069ms)
15:17:12 : EVENT : Connection Reconnected
message <Message 8>
15:17:13 : EVENT : Connection Broken
```

>> These messages are being placed into the 2nd queue manager QM_VER into queue SOURCE:

```
message <Message 9>
message <Message 10>
message <Message 11>
message <Message 12>
```

- Window-3 (get:amqsghac):

```
15:17:04 : EVENT : Connection Reconnecting (Delay: 9747ms)
15:17:13 : EVENT : Connection Reconnected
15:17:14 : EVENT : Connection Broken
```

>> These messages are being read from the queue SOURCE from the 2nd queue manager QM_VER

```
message <Message 8>
message <Message 9>
message <Message 10>
message <Message 11>
message <Message 12>
```

- To stop the samples, press <Ctrl-C> in each of the windows.

Scenario 3-C) Testing reconnect to another standalone queue manager, no sync point (using client configuration file)

This is a variation of Scenario 3-B. The main difference is that the CCDT file nor the MQSERVER variable will be used. Instead, the WebSphere MQ client configuration file will be used.

++ Edit the client configuration file in the local host (Host-A):

In the local Host-A (veracruz), modify the client configuration file:

```
/var/mqm/mqclient.ini
```

Or the file specified by the environment variable MQCLNTCF.

Add the following stanza (which will act as having the MQSERVER environment variable being set) and allowing the reconnectable clients to reconnect.

```
CHANNELS:
  DefRecon=YES
ServerConnectionParms=SYSTEM.DEF.SVRCONN/TCP/aemtux1.z.ibm.com(1421),veracr
uz.y.ibm.com(1414)
```

WARNING:
The format is: ChannelName/TransportType/ConnectionName where ConnectionName as a comma separated list of names of machines.
Do NOT include the single quotes around the queue managers. The single quotes are ok for MQSERVER in Unix, but it is NOT ok for the mqclient.ini.
Incorrect (using single quotes like in MQSERVER):

```
ServerConnectionParms=SYSTEM.DEF.SVRCONN/TCP/’aemtux1.z.ibm.com(1421),verac
ruz.y.ibm.com(1414)’
```

The above does NOT work and you will get the return code 2538 (MQRC_HOST_NOT_AVAILABLE):

```
MQCONNX ended with reason code 2538
```

- In each of the windows used in Scenario 3-C above, unset the environment variables that reference the CCDT file and MQSERVER:

```
unset MQCHLLIB
unset MQCHLTAB
unset MQSERVER
```

++ Stop the local queue manager and start the remote queue manager:

- Login to Host-A (veracruz):

```
        endmqm -i QM_VER
```

- Login to Host-B (aemtux1):

```
        strmqm ANGEL_TUX1
```

++ Start the samples in the local Host-A (veracruz).

++ Repeat the same steps as in Scenario 3-B

- In Window-2 (put: amqsphac):

```
$ amqsphac SOURCE
Sample AMQSPHAC start
target queue is SOURCE
message <Message 1>
```

- In Window-3 (get: amqsghac)
NOTICE that the queue is SOURCE (not TARGET)

```
$ amqsghac SOURCE
Sample AMQSGHAC start
message <Message 1>
```

- Wait for 15 seconds and observe the text in each window.

- In Window-5 of the remote Host-B, stop the queue manager and allow the clients to reconnect (flag: -r).

```
$ endmqm -r -i ANGEL_TUX1
```

- Observe the text in each window and verify that each program is trying to reconnect.

- Window-2 (put: amqsphac):

```
message <Message 6>
15:47:05 : EVENT : Connection Reconnecting (Delay: 134ms)
15:47:07 : EVENT : Connection Reconnecting (Delay: 0ms)
15:47:07 : EVENT : Connection Reconnecting (Delay: 2008ms)
15:47:09 : EVENT : Connection Reconnecting (Delay: 4768ms)
```

- Window-3 (get:amqsghac):

```
message <Message 6>
15:47:05 : EVENT : Connection Reconnecting (Delay: 43ms)
15:47:07 : EVENT : Connection Reconnecting (Delay: 0ms)
15:47:07 : EVENT : Connection Reconnecting (Delay: 2352ms)
15:47:10 : EVENT : Connection Reconnecting (Delay: 4078ms)
```

- In Window-1 of Host-A, restart the 2nd queue manager:

```
$ strmqm QM_VER
```

- Observe the text in each window and verify that each program reconnected.

- Window-2 (put: amqsphac):

```
15:47:09 : EVENT : Connection Reconnecting (Delay: 4768ms)
15:47:14 : EVENT : Connection Reconnected
message <Message 7>
15:47:14 : EVENT : Connection Broken
message <Message 8>
message <Message 9>
message <Message 10>
```

- Window-3 (get:amqsghac):

```
15:47:10 : EVENT : Connection Reconnecting (Delay: 4078ms)
15:47:14 : EVENT : Connection Reconnected
15:47:14 : EVENT : Connection Broken
message <Message 7>
message <Message 8>
message <Message 9>
message <Message 10>
```

- To stop the samples, press <Ctrl-C> in each of the windows.

++
+++ Chapter 4: Using the "High availability sample programs" (C code) – multi-instance queue managers
++

This is similar to Scenario 3-C):
<u>Testing reconnect to another standalone queue manager, no sync point (using client configuration file)</u>

The main difference is that instead of using 2 separate queue managers, a pair of multi-instance queue managers will be used (active and standby)

++ Edit the client configuration file in the local host (Host-A):

In the local Host-A (veracruz), modify the client configuration file:
```
        /var/mqm/mqclient.ini
```
Or the file specified by the environment variable MQCLNTCF.

Add the following stanza (which will act as having the MQSERVER environment variable being set) and allowing the reconnectable clients to reconnect.

```
CHANNELS:
  DefRecon=YES
ServerConnectionParms=SYSTEM.DEF.SVRCONN/TCP/cbeech.x.ibm.com(1421),veracru
z.y.ibm.com(1421)
```

WARNING:
The format is: ChannelName/TransportType/ConnectionName where ConnectionName as a comma separated list of names of machines.
Do NOT include the single quotes around the queue managers. The single quotes are ok for MQSERVER in Unix, but it is NOT ok for the mqclient.ini.
Incorrect (using single quotes like in MQSERVER):

```
ServerConnectionParms=SYSTEM.DEF.SVRCONN/TCP/'aemtux1.z.ibm.com(1421),verac
ruz.y.ibm.com(1414)'
```

The above does NOT work and you will get the return code 2538 (MQRC_HOST_NOT_AVAILABLE):
```
        MQCONNX ended with reason code 2538
```

Customization of the Client Channel Definition Table (CCDT)

Although we are not going to use the CCDT file in this scenario, this would be the changes to be made via runqmsc

This complements / replaces the runmqsc commands mentioned earlier in this document.

```
DEFINE CHANNEL(CHANNEL3) CHLTYPE(SVRCONN)  TRPTYPE(TCP) REPLACE

DEFINE CHANNEL(CHANNEL3) CHLTYPE(CLNTCONN)  TRPTYPE(TCP)  +
      CONNAME('cbeech.x.ibm.com(1421), +
veracruz.y.ibm.com(1421)') QMNAME(QMMI1) REPLACE

END
```

The location of the CCDT is:
/mqexport/701/data/QMMI1/@ipcc/AMQCLCHL.TAB

- In each of the windows used in Scenario 3-C above, unset the environment variables that reference the CCDT file and MQSERVER:

```
unset MQCHLLIB
unset MQCHLTAB
unset MQSERVER
```

++ Start the Multi-Instance queue managers:

- Login to Host-A (veracruz):

```
$ strmqm -x QMMI1
```

- Wait until this instance is started. It will be the Active one.

- Login to Host-C (cbeech):

```
$ strmqm -x QMMI1
```

- In Host-C, verify the status:

```
$ dspmq -x -m QMMI1
QMNAME(QMMI1)                                     STATUS(Running as standby)
   INSTANCE(veracruz) MODE(Active)
   INSTANCE(cbeech) MODE(Standby)
```

- In Host-A, verify the status:

```
$ dspmq -x -m QMMI1
QMNAME(QMMI1)                                        STATUS(Running)
    INSTANCE(veracruz) MODE(Active)
    INSTANCE(cbeech) MODE(Standby)\
```

- Stop the queue managers used in the previous scenarios.
In case that we have not done properly all the configuration changes, if these queue managers are running, they may give us a false reading.

- Login to Host-A (veracruz):

```
        endmqm -i  QM_VER
```

- Login to Host-B (aemtux1):

```
        endmqm -i ANGEL_TUX1
```

++ Start the samples in the local Host-A (veracruz).

- In Window-2 (put: amqsphac):

```
$ amqsphac SOURCE
Sample AMQSPHAC start
target queue is SOURCE
message <Message 1>
```

- In Window-3 (get: amqsghac)

```
$ amqsghac SOURCE
Sample AMQSGHAC start
message <Message 1>
```

- Wait for 15 seconds and observe the text in each window.

- In Window-1, stop the active instance of the queue manager and perform a switchover:

```
$ endmqm -s  -i QMMI1
Waiting for queue manager 'QMMI1' to end.
Waiting for queue manager 'QMMI1' to end.
Waiting for queue manager 'QMMI1' to end.
Waiting for queue manager 'QMMI1' to end.
```

```
Quiesce request accepted. The queue manager will stop when all outstanding work
is complete, permitting switchover to a standby instance.
```

- Observe the text in each window of the samples and verify that each program is trying to reconnect.

- Window-2 (put: amqsphac):

```
message <Message 6>
15:47:05 : EVENT : Connection Reconnecting (Delay: 134ms)
15:47:07 : EVENT : Connection Reconnecting (Delay: 0ms)
15:47:07 : EVENT : Connection Reconnecting (Delay: 2008ms)
15:47:09 : EVENT : Connection Reconnecting (Delay: 4768ms)
```

- Window-3 (get:amqsghac):

```
message <Message 6>
15:47:05 : EVENT : Connection Reconnecting (Delay: 43ms)
15:47:07 : EVENT : Connection Reconnecting (Delay: 0ms)
15:47:07 : EVENT : Connection Reconnecting (Delay: 2352ms)
15:47:10 : EVENT : Connection Reconnecting (Delay: 4078ms)
```

- In Window-6 of the remote Host-C, issue the following command to display the status.
Note: You may need to wait few seconds and then repeat the command, until you see that this standby instance is now the Active one.

```
$ dspmq -x -m QMMI1
QMNAME(QMMI1)                                    STATUS(Running)
    INSTANCE(cbeech) MODE(Active)
```

- Observe the text in each window and verify that each program reconnected.

- Window-2 (put: amqsphac):

```
15:47:09 : EVENT : Connection Reconnecting (Delay: 4768ms)
15:47:14 : EVENT : Connection Reconnected
message <Message 7>
15:47:14 : EVENT : Connection Broken
message <Message 8>
message <Message 9>
message <Message 10>
```

- Window-3 (get:amqsghac):

```
15:47:10 : EVENT : Connection Reconnecting (Delay: 4078ms)
15:47:14 : EVENT : Connection Reconnected
```

```
15:47:14 : EVENT : Connection Broken
message <Message 7>
message <Message 8>
message <Message 9>
message <Message 10>
```

- To stop the samples, press <Ctrl-C> in each of the windows.

++
+++ Chapter 5: Behavior of clients that are not explicitly enabled for automatic reconnect (using client configuration file)
++

The sample “amqsputc” (put message) does not have source code that explicitly enable the handling of the automatic reconnect.

The idea in these 2 Scenarios is to show you what the behavior is when the queue manager is ending and is asking the connected clients to reconnect.

5-A: Using amqsputc that was compiled and linked with MQ 7.0.1 and running under MQ 7.0.1

5-B: Using amqsputc that was compiled and linked with MQ 6 and was copied into a machine that is running MQ 7.0.1.

These samples use the underlying MQI calls from the MQ dynamically-linked libraries provided in MQ 7.0.1 and thus, these MQI calls have a default behavior of trying to reconnect and to follow the instructions provided in the Client Configuration File (mqclient.ini).

Note about not using amqsgetc:
The sample “amqsgetc” is too limiting for this scenario because it terminates after 30 seconds of inactivity, and in these scenarios, it takes more than 30 seconds do to the switchover. Thus, the high availability sample “amqsghac” will be used instead, and there is an advantage that we can observe the messages that indicate the reconnection.

<u>Scenario 5-A) Testing reconnect to another standalone queue manager, amqsputc and amqsgetc (C-code), MQ 7.0.1</u>

The scenario is similar to the scenario in Chapter 4, in which Multi-Instance queue managers are specified for the automatic client reconnect.

The samples are the ones provided with MQ 7.0.1, that is, the source code for the samples was compiled and linked under MQ 7.0.1.

The sample amqsputc was compiled and linked with MQ 7.0.1 and is running under MQ 7.0.1.

- Unset the environment variables that reference the CCDT file and MQSERVER:
unset MQCHLLIB

```
unset MQCHLTAB
unset MQSERVER
```

++ Ensure that the local client configuration file is properly set for 2 standalone queue managers:

```
$ tail /var/mqm/mqclient.ini

CHANNELS:
  DefRecon=YES
ServerConnectionParms=SYSTEM.DEF.SVRCONN/TCP/cbeech.x.ibm.com(1421),veracru
z.y.ibm.com(1421)
```

++ Start the Multi-Instance queue managers:

- Login to Host-A (veracruz):

```
$ strmqm -x QMMI1
```

- Wait until this instance is started. It will be the Active one.

- Login to Host-C (cbeech):

```
$ strmqm -x QMMI1
```

++ Start the samples in the local Host-A (veracruz).

- In Window-2 (put: amqsputc).

Enter 3 short strings followed by Enter for each one: test1, test2 and test3

```
$ amqsputc SOURCE
Sample AMQSPUT0 start
target queue is SOURCE
test1
test2
test3
```

- In Window-3 (get: amqsghac)

```
$ amqsgetc SOURCE
Sample AMQSGHAC start
message <test1>
message <test2>
message <test3>
```

- Wait for 15 seconds and observe the text in each window.

- In Window-1, stop the active instance of the queue manager and perform a switchover:

```
$ endmqm -s QMMI1
```

- Observe the text in each window of the samples and verify that each program is trying to reconnect.

- Window-2 (put: amqsputc).
Enter 3 more messages:

```
Sample AMQSPUT0 start
target queue is SOURCE
test1
test2
test3
test4
test5
test6
```

- Window-3 (get:amqsputc):

```
Sample AMQSGHAC start
message <test1>
message <test2>
message <test3>
14:39:46 : EVENT : Connection Reconnecting (Delay: 172ms)
14:39:46 : EVENT : Connection Reconnecting (Delay: 1311ms)
14:39:47 : EVENT : Connection Reconnecting (Delay: 2213ms)
14:39:50 : EVENT : Connection Reconnecting (Delay: 4522ms)
14:39:54 : EVENT : Connection Reconnecting (Delay: 8654ms)
14:40:03 : EVENT : Connection Reconnecting (Delay: 18782ms)
14:40:22 : EVENT : Connection Reconnecting (Delay: 29246ms)
14:40:51 : EVENT : Connection Reconnecting (Delay: 33824ms)
```

- In Window-6 of the remote Host-C, issue the following command to display the status.
Note: You may need to wait few seconds and then repeat the command, until you see that this standby instance is now the Active one.

```
$ dspmq -x -m QMMI1
QMNAME(QMMI1)                                    STATUS(Running)
   INSTANCE(cbeech) MODE(Active)
```

- Window-2 (put: amqsphac):

Enter 3 more messages:

```
Sample AMQSPUT0 start
target queue is SOURCE
test1
test2
test3
test4
test5
test6
test7
test8
test9
```

- Window-3 (get:amqsghac):

Notice that the messages "test4" thru "test6" where issued while there was a switchover, and the rest of the messages were issued after the switchover.

```
Sample AMQSGHAC start
message <test1>
message <test2>
message <test3>
14:39:46 : EVENT : Connection Reconnecting (Delay: 172ms)
14:39:46 : EVENT : Connection Reconnecting (Delay: 1311ms)
14:39:47 : EVENT : Connection Reconnecting (Delay: 2213ms)
14:39:50 : EVENT : Connection Reconnecting (Delay: 4522ms)
14:39:54 : EVENT : Connection Reconnecting (Delay: 8654ms)
14:40:03 : EVENT : Connection Reconnecting (Delay: 18782ms)
14:40:22 : EVENT : Connection Reconnecting (Delay: 29246ms)
14:40:51 : EVENT : Connection Reconnecting (Delay: 33824ms)
14:41:25 : EVENT : Connection Reconnecting (Delay: 29937ms)
14:41:55 : EVENT : Connection Reconnected
14:41:55 : EVENT : Connection Broken
message <test4>
message <test5>
message <test6>
message <test7>
```

message <test8>
message <test9>

- To stop the samples, press <Ctrl-C> in each of the windows.

Scenario 5-B) Testing reconnect to another standalone queue manager, amqsputc and amqsgetc (C-code), MQ 6

The scenario is similar to the Scenario 5-A.

The main difference is that the sample amqsputc was compiled and linked with MQ 6.0.2.8 and it was copied into a host that is running under MQ 7.0.1.

Note:
The scenario 5-B behaved exactly as in Scenario 5-A.
It proves that when a client that has been dynamically linked in MQ V6 and copied into a host that uses the MQ client 7.0.1, it will behave with the default automatic client reconnect.

++
+++ Appendix A: Troubleshooting
++

+++ A) If a client application is running without a specific queue manager name and the environment variables for the CCDT or MQSERVER are not defined, and the client configuration file is not set (or not configured properly), and there is no default queue manager, then the following error message is displayed:

```
$ unset MQCHLLIB
$ unset MQCHLTAB
$ unset MQSERVER

$ amqsphac SOURCE
Sample AMQSPHAC start
MQCONNX ended with reason code 2538
Sample AMQSPHAC end

$ mqrc 2538
     2538  0x000009ea  MQRC_HOST_NOT_AVAILABLE
```

+++ B) Specifying the hostname(port) in MQSERVER

B.1) In Unix, it is necessary to use single quotes around hostname(port):

```
  export MQSERVER=SYSTEM.DEF.SVRCONN/TCP/'aemtux1.z.ibm.com(1421)'
```

or around the entire string:

```
  export MQSERVER='SYSTEM.DEF.SVRCONN/TCP/aemtux1.z.ibm.com(1421)'
```

B.2) However, it might be confusing to see how the Bourne, Korn or Bash shells display the value:

```
$ export MQSERVER=SYSTEM.DEF.SVRCONN/TCP/'aemtux1.z.ibm.com(1421)'

# Notice that the output of echo does not show any single quotes
$ echo $MQSERVER
SYSTEM.DEF.SVRCONN/TCP/aemtux1.z.ibm.com(1421)
rivera@veracruz: /home/rivera

# Notice that the single quotes surround the entire string, not just hostname(port)
$ set | grep MQSERVER
MQSERVER='SYSTEM.DEF.SVRCONN/TCP/aemtux1.z.ibm.com(1421)'
```

B.3) If the single quotes are not used, then the following error will occur:

```
$ export MQSERVER=SYSTEM.DEF.SVRCONN/TCP/aemtux1.z.ibm.com(1421)
-bash: syntax error near unexpected token `('
```

B.4) In Windows, there is no need to use the single quotes around hostname(port):

B.4.a) Correct:

```
C:\> set MQSERVER=SYSTEM.DEF.SVRCONN/TCP/aemtux1.z.ibm.com(1421)

C:\> set MQSERVER
MQSERVER=SYSTEM.DEF.SVRCONN/TCP/aemtux1.z.ibm.com(1421)

C:\> amqsputc SOURCE
Sample AMQSPUT0 start
target queue is SOURCE
test from Windows
Sample AMQSPUT0 end
```

B.4.b) Incorrect: using single quotes around hostname(port)

```
C:\> set MQSERVER=SYSTEM.DEF.SVRCONN/TCP/'aemtux1.z.ibm.com(1421)'

C:\> amqsputc SOURCE
Sample AMQSPUT0 start
MQCONN ended with reason code 2538

C:\> mqrc 2538
     2538  0x000009ea  MQRC_HOST_NOT_AVAILABLE
```

B.4.c) Incorrect: using single quotes entire string

```
C:\> set MQSERVER='SYSTEM.DEF.SVRCONN/TCP/aemtux1.z.ibm.com(1421)'

C:\> amqsputc SOURCE
Sample AMQSPUT0 start
MQCONN ended with reason code 2540

C:\> mqrc 2540
     2540  0x000009ec  MQRC_UNKNOWN_CHANNEL_NAME
```

+++ C) Do not use single quotes around the connection name in the client configuration file

The following is a correct way to specify the Channels stanza in mqclient.ini:

```
CHANNELS:
DefRecon=YES
ServerConnectionParms=SYSTEM.DEF.SVRCONN/TCP/aemtux1.z.ibm.com(1421)
```

Note: Do NOT use single quotes around the connection name! The MQSERVER variable in Unix requires them, but they do NOT work inside the client configuration file.

```
  ServerConnectionParms=SYSTEM.DEF.SVRCONN/TCP/'aemtux1.z.ibm.com(1421)'
```

Or in the case of multiple hostname(port) tokens:

```
ServerConnectionParms=SYSTEM.DEF.SVRCONN/TCP/’aemtux1.z.ibm.com(1421),verac
ruz.y.ibm.com(1414)’
```

The above does NOT work and you will get the return code 2538 (MQRC_HOST_NOT_AVAILABLE):

```
   MQCONNX ended with reason code 2538
```

+++ D) Applications that use sync point can perform automatic client reconnect only for the same queue manager.

If a reconnectable application that uses sync point tries to reconnect to another queue manager, then the reason code 2548 is generated:

```
15:40:21 : EVENT : Connection Reconnecting (Delay: 4747ms)
15:40:26 : EVENT : Reconnection failed
MQGET ended with reason code 2548
MQBACK ended with reason code 2548
MQDISC ended with reason code 2548
Sample AMQSMHA0 end
```

```
$ mqrc 2548
     2548  0x000009f4  MQRC_RECONNECT_FAILED
```

+++ E) An application that does not specify the automatic reconnect, then will not be able to reconnect if the queue manager restarts:

```
$ amqsgetc SOURCE

MQGET ended with reason code 2009
MQCLOSE ended with reason code 2009
MQDISC ended with reason code 2009
Sample AMQSGET0 end

$ mqrc 2009
     2009  0x000007d9  MQRC_CONNECTION_BROKEN
```

+++ F) You have a CCDT with several channels and there is no default queue manager for the host. You start the amqsphac sample and do not specify the queue manager name.

```
$ amqsphac SOURCE
Sample AMQSPHAC start
MQCONNX ended with reason code 2539
Sample AMQSPHAC end

$ mqrc 2539
     2539  0x000009eb  MQRC_CHANNEL_CONFIG_ERROR
```

This error generates an entry for AMQ9205 in the /var/mqm/errors MQ error log.

```
01/26/2010 12:26:48 PM - Process(29467.1) User(rivera) Program(amqsphac)
                 Host(veracruz)
AMQ9205: The host name supplied is not valid.
EXPLANATION:
The supplied TCP/IP host name '' could not be resolved into a network address.
Either the name server does not contain the host, or the name server was not
available.
ACTION: Check the TCP/IP configuration on your host.
```

Solution:
Because the CCDT contains multiple channel entries and because there is no default queue manager, then the MQ client application does not know which queue manager to contact.
The solution is to specify the name of the queue manager when invoking the sample:

```
        $ amqsphac SOURCE QM_VER
```

+++
+++ Appendix B: Information about the MQ Client Configuration File
+++

The MQ Client Configuration File was introduced in MQ V7.

Its functionality was extended in MQ 7.0.1 to incorporate the automatic client reconnect feature.

For more information consult:
http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp?topic=/com.ibm.mq.csqzaf.doc/cs13350_.htm
WebSphere MQ client configuration file
“
The configuration features apply to all connections a client application makes to any queue managers, rather than being specific to an individual connection to a queue manager. Attributes relating to a connection to an individual queue manager can be configured programmatically, for example by using an MQCD structure, or by using a Client Channel Definition Table (CCDT).
“

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp?topic=/com.ibm.mq.csqzaf.doc/cs13360_.htm
Location of the client configuration file

The default file name is:
    mqclient.ini

The default location is:
On UNIX systems, the directory is:
    /var/mqm

On Windows® platforms you configure the environment variable MQ_FILE_PATH during installation, to point at the installation directory. It is normally:
    C:\Program Files\IBM\WebSphere MQ

The file can reside in a different directory or have a different name. The name and location will be the specified by the environment variable:
    MQCLNTCF

The configuration changes needed for the reconnect scenarios need to reside in the CHANNELS stanza of the client configuration file. For more details consult:
http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp?topic=/com.ibm.mq.csqzaf.doc/cs13370_.htm
CHANNELS stanza of the client configuration file

The following is a summary of the important variables:

ChannelDefinitionDirectory=path
The directory path to the file containing the client channel definition table.
On Windows the default is the WebSphere MQ installation directory, typically C:\Program Files\IBM\WebSphere MQ: on Unix systems the default is /var/mqm.
This is equivalent to the MQCHLLIB environment parameter.

ChannelDefinitionFile=filename|AMQCLCHL.TAB
The name of the file containing the client channel definition table.
This is equivalent to the MQCHLTAB environment parameter.

DefRecon=NO|YES|QMGR|DISABLED
The DefRecon attribute provides an administrative option to enable client programs to automatically reconnect, or to disable the automatic reconnection of a client program that has been written to reconnect automatically.

NO - Unless overridden by MQCONNX, the client is not reconnected automatically.
YES - Unless overridden by MQCONNX, the client reconnects automatically.
QMGR - Unless overridden by MQCONNX, the client reconnects automatically, but only to the same queue manager. The QMGR option has the same effect as MQCNO_RECONNECT_Q_MGR.
DISABLED - Reconnection is disabled, even if requested by the client program using the MQCONNX MQI call.

MQReconnectTimeout=TimeInSeconds
The timeout in seconds for retrying a client reconnection. The default value is 1800 seconds (30 minutes).

ServerConnectionParms
ServerConnectionParms specifies the location of the WebSphere MQ server and the communication method to be used. It is a string of the format ChannelName/TransportType/ConnectionName.
ConnectionName must be a fully-qualified network name.
ChannelName cannot contain the forward slash (/) character because this character is used to separate the channel name, transport type, and connection name.

When ServerConnectionParms is used to define a client channel, a maximum message length of 100 MB is used. Therefore the maximum message size in effect for the channel is the value specified in the SVRCONN channel on the server.
This is equivalent to the MQSERVER environment parameter.

Specify ConnectionName as a comma separated list of names of machines for the stated TransportType. Typically, only one machine name is required. You can provide multiple machine names to configure multiple connections with the same properties. The connections are tried in the order they are specified in the connection list until a connection is successfully established. If no connection is successful, the client starts retry processing. Connection lists are an alternative to queue manager groups to configure connections for reconnectable clients.

+++ end +++