IBM Software Group

Exploiting Automatic Client Reconnect in WebSphere MQ 7

Angel Rivera (rivera@us.ibm.com)
WebSphere MQ Unix® and Windows Level 2 Support
Date: 01-May-2013

ON DEMAND BUSINESS

Agenda

- Introduction
- Simple scenario using non-reconnectable client (using MQSERVER)
- Simple scenario using reconnectable client
- Precedence of mechanisms for a client to connect to a queue manager
- Reconnect interval (using mqclient.ini)
- Reconnection to another queue manager, using connectionNameList
- Scenario using CCDT
- MQ JMS client from WAS

What is automatic client reconnect?

- Allows MQ client applications to reconnect automatically, without writing any additional code, by configuring a number of components.
  - In some cases, you will need to modify the application to incorporate new options.
- Automatic client reconnection is inline.
  - The connection is automatically restored at any point in the client application program, and the handles to open objects are all restored.

Only for Transport Type of Client

- This feature applies to MQ clients connecting to the queue manager using a transport type of:
  - CLIENT
- That is, using a network connection via a
  - Server-Connection Channel (SVRCONN)

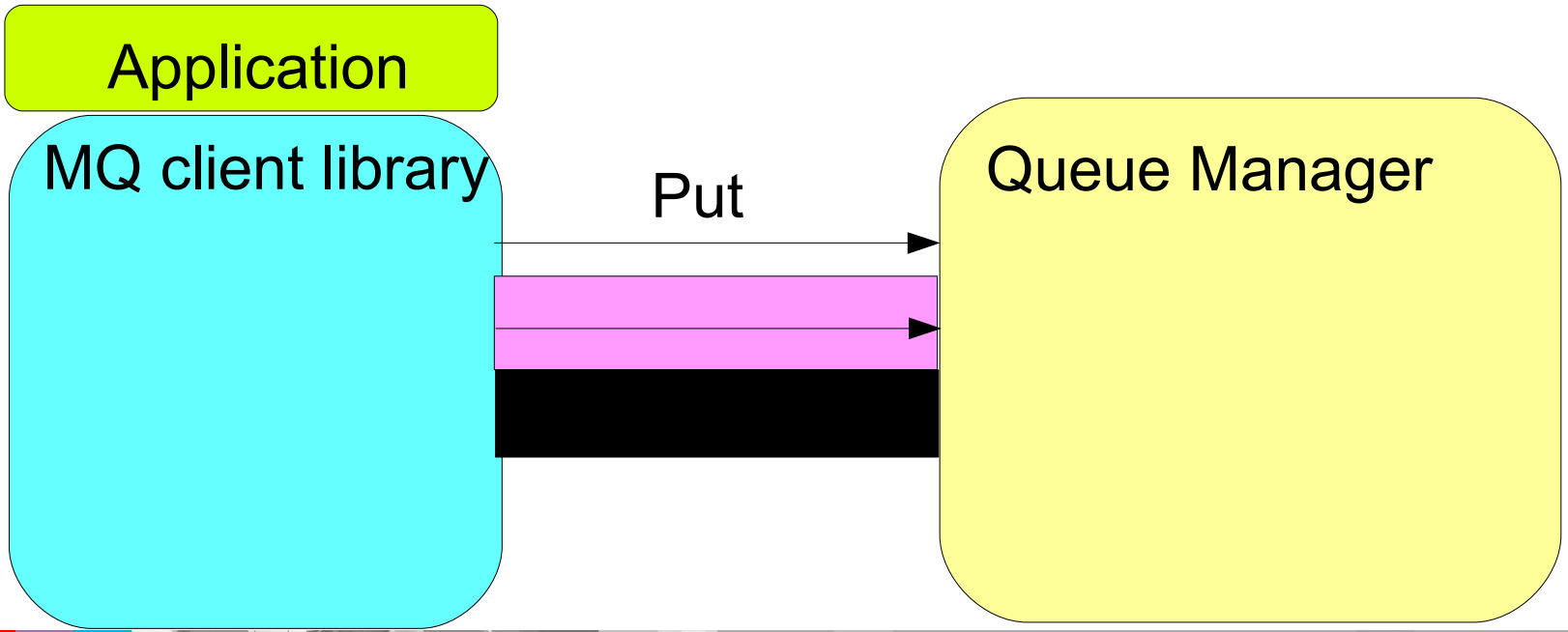
- It does NOT apply to MQ clients using a transport type of:
  - BINDINGS (inter process communications)

Requirements

- Both the Client application and the Queue Manager must be at: 7.0.1.0 or higher
  - JMS is 7.0.1.3 or higher.
- Server-Connection channel must be Full-Duplex:
- The “shared conversations” of the channel attribute MUST have a value of 1 or greater:
  - SHARECNV equal or greater than 1
- A value of 0 means the channel is Half-Duplex.
- A value of 1 or greater, means Full-Duplex

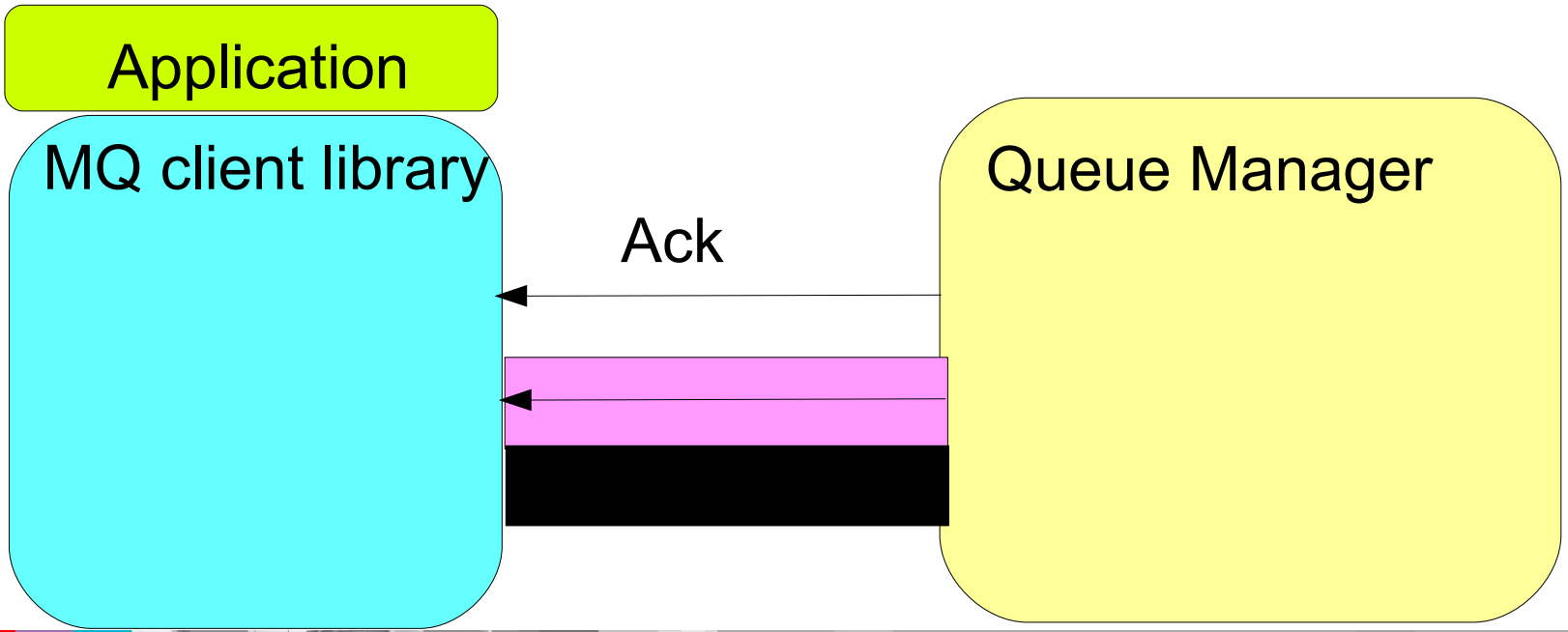
Half-duplex – very simple explanation (1)

- When using Half-Duplex, is like using a 2 lane road but only 1 lane is open for both data and control (the other lane is closed)



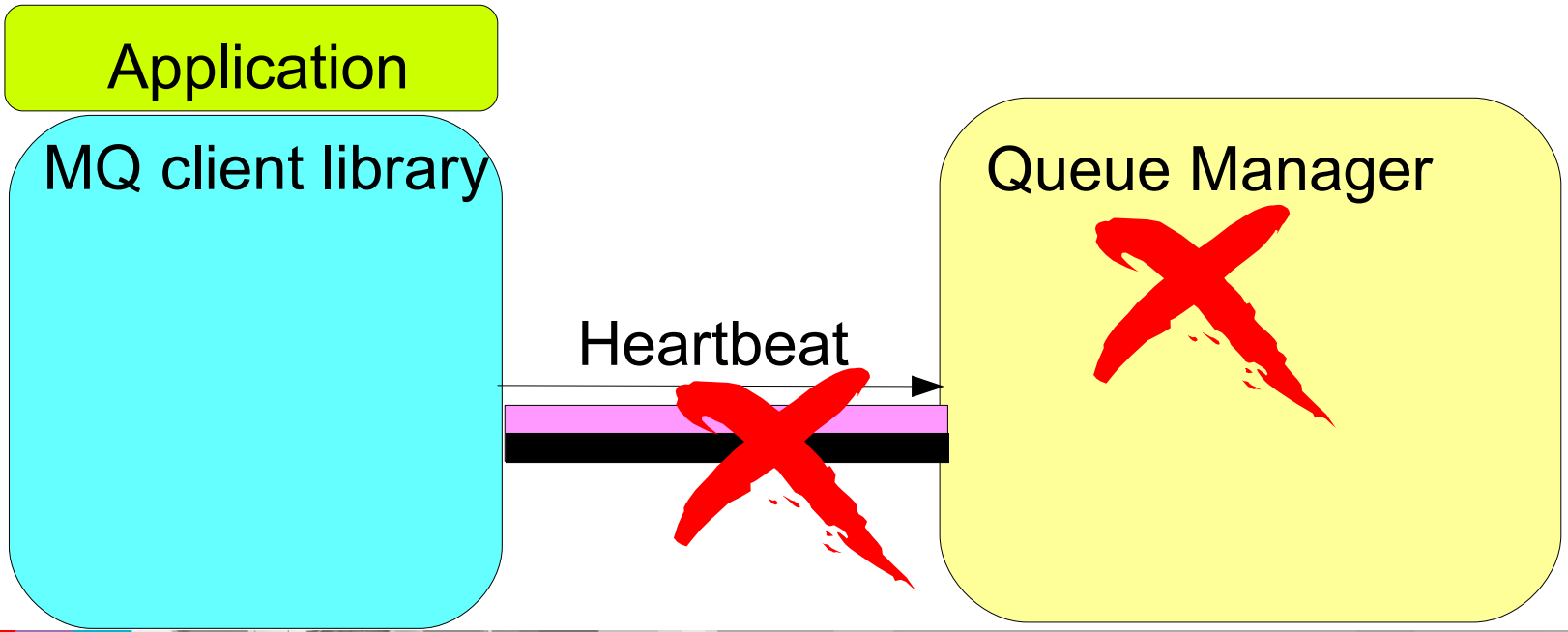
Half-duplex – very simple explanation (2)

- The queue manager received the Put request and then replies with an acknowledgement to the client application



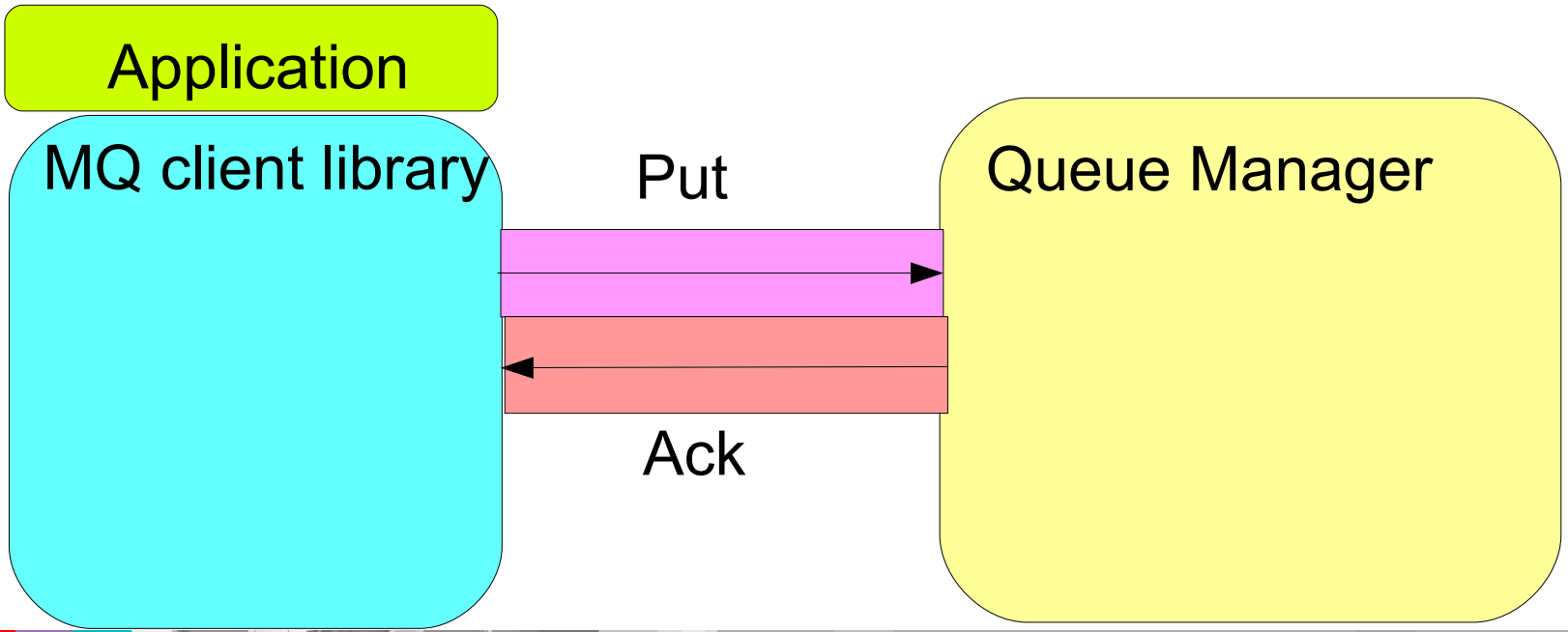
Half-duplex – very simple explanation (3)

- If the network has a problem or if the queue manager ends unexpectedly, the MQ client CANNOT send a heartbeat to the queue manager asking for status.



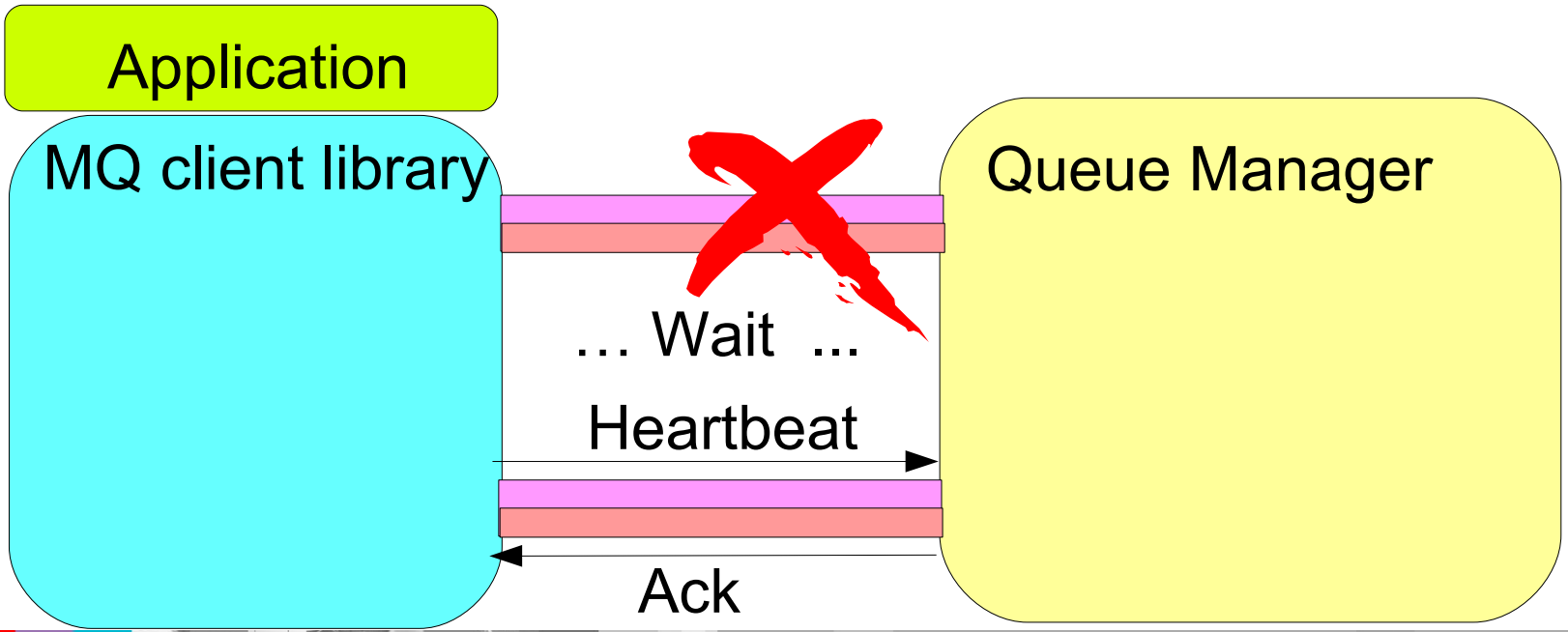
Full-duplex – very simple explanation (1)

- When using Full-Duplex, is like using a 2 lane road AND both lanes are open for both data and control



Full-duplex – very simple explanation (2)

- Can send heartbeat to see if the other component is still connected!



What components do NOT support ACR

- The following components do NOT support ACR:
  - WebSphere MQ classes for Java
  - Managed XMS and managed .NET clients: C#, Visual Basic

- See table 1 “Supported clients”:
- http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/topic/com.ibm.mq.doc/fa70190_.htm
- WebSphere MQ > Configuring > Availability, recovery and restart
- Automatic client reconnection

What about XA?

- When a failure occurs ‘during’ a transaction, the transaction is ‘doomed’ – final MQCMIT will always back out:
- MQRC_BACKED_OUT returned to application
- Non-transactional tasks are allowed to complete

- XA and reconnection to multiple / different queue managers are mutually exclusive
- The XA interface is too restrictive to guarantee data integrity when dealing with multiple / different queue managers.

Notes: XA

notes

- http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/topic/com.ibm.iea.wmq_v7/wmq/7.0.1/Details/iea_701_130_recon_client.pdf
- XA and reconnection [to different queue managers] are mutually exclusive
- – The XA interface is too restrictive to guarantee data integrity between multiple resource managers if a connection to MQ is re-established silently

- http://www.ibm.com/support/docview.wss?uid=swg21508357
- Using WebSphere MQ automatic client reconnection with the WebSphere MQ classes for JMS
- If the Activation Specifications are being used with transactional message-driven beans that participate in XA transactions, and are connecting to a multi-instance queue manager, then the CONNECTIONNAMELIST must contain 2 entries:
- An entry for the Active queue manager instance.
- An entry for the Standby queue manager instance.
- If the transactional message-driven beans are being used with stand-alone queue managers, then the CONNECTIONNAMELIST property must contain a single entry, to ensure that the Activation Specification always reconnects to the same queue manager running on the same system following a failure.
-

What is a reconnectable scenario ?

- Only explicit ends or failures
  - Communications failure
  - Queue Manager or Listener failure
  - STOP CONN
  - endmqm –s or endmqm –r
- The following will not cause reconnect
  - STOP CHANNEL
  - Any other endmqm

Notes: reconnect related options for endmqm

notes

- http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/topic/com.ibm.mq.doc/fa15800_.htm
- WebSphere MQ > Reference > Administration reference > WebSphere MQ control commands > The control commands > endmqm
- **-i Immediate shutdown.**
- The queue manager stops after it has completed all the MQI calls currently being processed. Any MQI requests issued after the command has been issued fail. Any incomplete units of work are rolled back when the queue manager is next started.
- Control is returned after the queue manager has ended.
- The effect on any client applications connected through a server-connection channel is equivalent to a STOP CHANNEL command issued in FORCE mode.
- **-r Start trying to reconnect reconnectable clients.**
- This parameter has the effect of reestablishing the connectivity of clients to other queue managers.
- **-s Switch over and start trying to reconnect reconnectable clients.**
- Switch over to a standby queue manager instance after shutting down. The command checks that there is a standby instance running before ending the active instance. It does not wait for the standby instance to start before ending.
- Connections to the queue manager are broken by the active instance shutting down. Reconnectable clients start trying to reconnect.

Notes: samples to put and get messages

notes

Baseline scenario uses amqsputc, which is part of the traditional samples to put and get messages:

- amqsput – put message, bindings mode
- amqsget – get message, bindings mode
- **amqsputc – put message, client mode**
- amqsgetc – get message, client mode

- They are located at:

  Windows:

  Execs: C:\Program Files\IBM\WebSphere MQ\tools\c\Samples\Bin

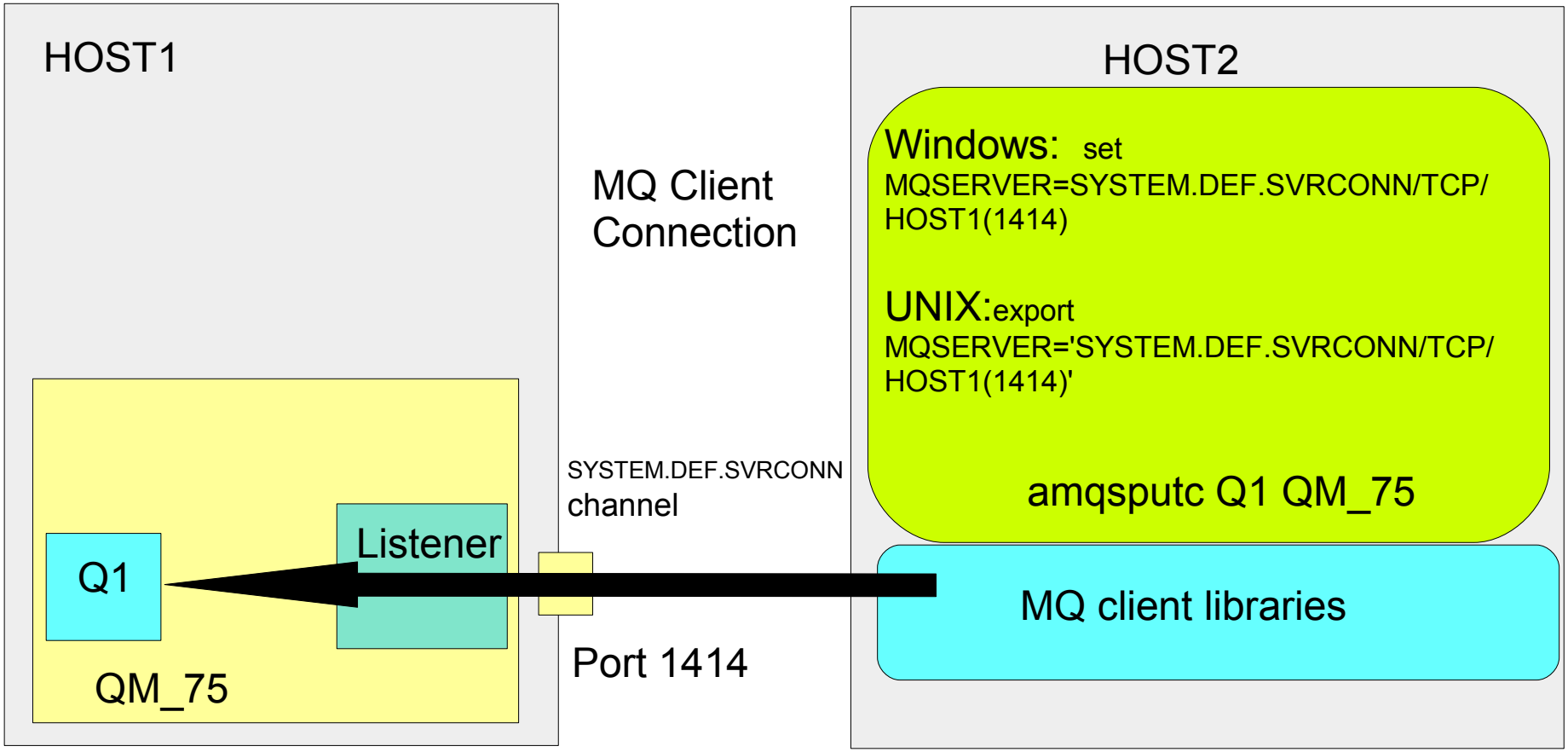
  Source: C:\Program Files\IBM\WebSphere MQ\tools\c\Samples
- UNIX: Executables: /opt/mqm/samp/bin

  Source: /opt/mqm/samp

Baseline scenario - amqsputc

- Let's start with a basic scenario.
- Using the MQ client application “amqsputc” which connects to the queue manager in Client mode (not bindings) and puts 1 message into a queue
- After the application connects and puts 1 message, then the queue manager is terminated with “endmqm -ir” requesting clients to reconnect.
- For example, need to restart the queue manager to install an interim fix.

Baseline scenario - diagram



Baseline scenario (1)

- Establish how to connect to queue manager:
- Windows:
- set MQSERVER=SYSTEM.DEF.SVRCONN/TCP/HOST1(1414)
- UNIX:
- export MQSERVER='SYSTEM.DEF.SVRCONN/TCP/HOST1(1414)'

- MQ client application (sample amqsputc) connects to queue manager and puts messages.
- amqsputc Q1 QM_75
- Sample AMQSPUT0 start
- target queue is Q1
- message-1
- (pausing queue manager – see next slide)

Baseline scenario (2)

- At this point, the queue manager is terminated:
  endmqm -ir QM_75
- -i indicates to end immediately
- -r indicates that reconnectable clients should try to reconnect (to be explained in detail later)

- If you try to continue to use amqsputc, you will see:
  - MQPUT ended with reason code 2009
  - MQCLOSE ended with reason code 2009
  - MQDISC ended with reason code 2009
  - Sample AMQSPUT0 end

Baseline scenario (3)

- The following error will be generated at the General error log:
- Program(amqsputc.exe)
- Host(ANGELITO) Installation(Installation2)
- VRMF(7.5.0.0)
- AMQ9208: Error on receive from host localhost (127.0.0.1)(1439).
- EXPLANATION:
- An error occurred receiving data from localhost (127.0.0.1)(1439) over TCP/IP. This may be due to a communications failure.
- ACTION:
- The return code from the TCP/IP recv() call was 10054 (X'2746'). Record these values and tell the systems administrator.

- **Note:** 10054 WSAECONNRESET -- Connection reset by peer. This occurs when an established connection is shut down for some reason by the remote computer.

Baseline scenario (4)

- The MQ client application (sample amqsputc) was not developed to take advantage of the automatic client reconnection
- The application could not begin to try to reconnect when the queue manager ended gracefully and requested the reconnectable clients to try to reconnect.
- When the queue manager restarted, the application was NOT able to reconnect.

Scenario using amqsphac (1)

- If you want to test the automatic reconnection feature you will need to use additional samples.

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

- amqsphac - put message, client mode
- amqsghac - get message, client mode

Scenario using amqsphac (2)

- Let's perform the same steps as in the basic scenario, but this time, let's use the MQ client application “amqsphac” which connects to the queue manager in Client mode (not bindings) and puts messages into a queue

- After the application connects and puts several messages, then the queue manager is terminated with “endmqm -ir” requesting clients to reconnect.

Scenario using amqsphac (3)

- Setting same MQSERVER variable
- c:\> set MQSERVER=SYSTEM.DEF.SVRCONN/TCP/HOST1(1414)

- amqsphac Q1
- Sample AMQSPHAC start
- target queue is Q1
- message <Message 1>
- message <Message 2>
- (pausing queue manager – see next slide)

Scenario using amqsphac (4)

- At this point, the queue manager is terminated:
  endmqm -ir QM_75

- amqsphac identifies the end of the queue manager and was asked to reconnect:
- message <Message 3>
- 10:29:48 : EVENT : Connection Reconnecting (Delay: 66ms)
- 10:29:48 : EVENT : Connection Reconnecting (Delay: 989ms)
- 10:29:51 : EVENT : Connection Reconnecting (Delay: 1399ms)

Scenario using amqsphac (5)

- The queue manager is restarted:
  strmqm QM_75

- amqsphac identifies that the queue manager is running again and reconnects:

- 10:29:53 : EVENT : Connection Reconnecting (Delay: 3415ms)
- 10:29:57 : EVENT : Connection Reconnecting (Delay: 7357ms)
- 10:30:05 : EVENT : Connection Reconnected
- message <Message 4>
- message <Message 5>

Notes: RC 2162 if -r / -s not used in endmqm

notes

- If the queue manager is ended WITHOUT the proper flags:
- reconnection (-r) or switchover (-s)
- then even reconnectable clients will terminate and will not reconnect when the queue manager restarts.

- In this example, the queue manager was ended with: endmqm -i Qmgr

- C:> amqsphac Q1
- Sample AMQSPHAC start
- target queue is Q1
- message <Message 1>
- MQPUT ended with reason code 2162 10:45:32 : EVENT : Reason(2162)
- MQCLOSE ended with reason code 2009 10:45:32 : EVENT : Connection Broken
- Sample AMQSPHAC end

- C:> mqrc 2162
- MQRC_Q_MGR_STOPPING

Precedence for connecting to a QMgr

- This is the PRECEDENCE of the mechanisms provided by the MQ client for an application to connect to a queue manager.

  1: Pre-connect exit (added in MQ 7.1 - advanced)
  2: MQCONN API call (in the application's code)
  3: MQSERVER environment variable
  4: mqclient.ini file (introduced in MQ 7.0)
  5: CCDT file (LOWEST precedence)

Unset MQSERVER environment variable

- In the previous scenarios we did not use:
- 1: Pre-connect exit (added in MQ 7.1 – advanced)
- 2: MQCONN API call (in the application's code)

- Instead, we used:
- 3: MQSERVER environment variable

- We need to unset MQSERVER in order to use:
- 4: mqclient.ini file (introduced in MQ 7.0)
- 5: CCDT file (LOWEST precedence)

MQ Client configuration file: mqclient.ini

- MQ Client configuration file is called by default:
- mqclient.ini

- All MQ client applications that can access the above file can use the settings within it.

- This presentation only covers the stanza:
  CHANNELS

- The location of this file is described in the following slide.

Notes: location of mqclient.ini

notes

- http://pic.dhe.ibm.com/infocenter/wmqv7/v7r5/topic/com.ibm.mq.doc/cs13360_.htm
- WebSphere MQ > Configuring > Configuring connections between the server and clients > Configuring a client using a configuration file
- Location of the client configuration file

- A client application uses the following search path to locate the WebSphere MQ MQI client configuration file:
- 1. The location specified by the environment variable MQCLNTCF.
- 2. A file called mqclient.ini in the present working directory of the application.
- 3. A file called mqclient.ini in the WebSphere MQ data directory for Windows, UNIX and Linux systems.
- - On UNIX and Linux systems, the directory is /var/mqm
- - On Windows platforms you configure the environment variable MQ_FILE_PATH during installation, to point at the data directory. It is normally C:\Program Files\IBM\WebSphere MQ
- 4. A file called mqclient.ini in a standard directory appropriate to the platform, and accessible to users.

Using mqclient.ini

- Let's repeat the scenario with amqsphac.
- We need to unset MQSERVER to use mqclient.ini
- Windows: set MQSERVER=
- UNIX: unset MQSERVER
- In this scenario, this file is located in the data path, which is specified by the variable: MQ_DATA_PATH
- The following environment variable can be used to specify the location: MQCLNTCF

Using mqclient.ini – Channels stanza

- Example of the contents:

- CHANNELS:
- DefRecon=YES
- ServerConnectionParms=SYSTEM.DEF.SVRCONN/TCP/localhost(1414)

- Notice that the specification is like in Windows, without single quotes.
- Even when using mqclient.ini in a UNIX machine.

Reconnect interval

- Maximum Reconnect Interval
- Defaults to 1,800 seconds (30 minutes)
- Setting in mqclient.ini: **MQReconnectTimeout**
- Reconnect Intervals
- Delay consists of fixed and random part
- Can be changed in mqclient.ini
- **ReconDelay** = (1000,200) (2000,200) (4000,1000)

- Initial delay 1 sec + random interval of up to 200 ms
- 2nd delay is 2 secs + up to 200 ms
- Later delays are 4 secs + up to 1000 ms

Reconnect interval (2)

- The main reason for having a combination of a fixed and random part is to avoid a massive reconnect attempt simultaneously from many clients.
- For example, there are 1,000 reconnectable client connected to a queue manager. The queue manager is stopped with -ir and restarted.
- If there was only a fixed part in the reconnect interval, then the 1,000 clients will try to reconnect at the same time (kind of Denial of Service attack!), potentially overloading the MQ listener.
- But having a random part allows for spreading the reconnection attempts without overloading.

Notes: RC 2548 when max recon interval

notes

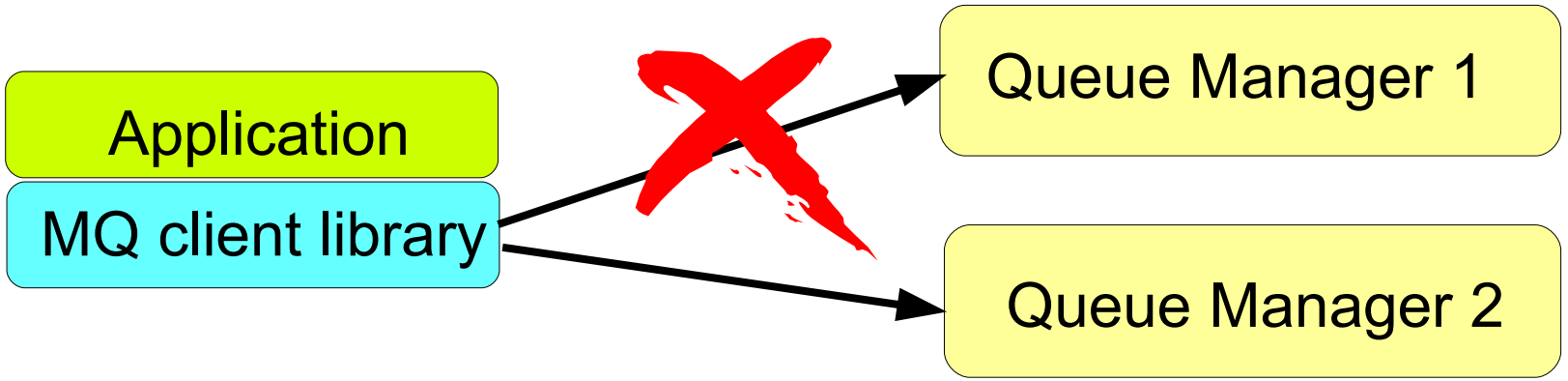
- If the queue manager is ended with the proper flags:
- reconnection (-r) or switchover (-s)
- But it is not restarted within the maximum reconnect interval (default is 30 minutes), then the reconnectable clients will issue RC 2548 and terminate:

- C:\> amqsphac Q1
- Sample AMQSPHAC start
- target queue is Q1
- message <Message 1>
- message <Message 2>
- 10:56:47 : EVENT : Connection Reconnecting (Delay: 7ms)
- 10:56:47 : EVENT : Connection Reconnecting (Delay: 941ms)
- ...
- 10:58:24 : EVENT : Connection Reconnecting (Delay: 33292ms)
- 10:58:24 : EVENT : Reconnection failed
- 10:58:24 : EVENT : Connection Broken
- MQPUT ended with reason code 2548
- MQCLOSE ended with reason code 2548
- Sample AMQSPHAC end

- mqrc 2548 => **MQRC_RECONNECT_FAILED**

Reconnection to another queue manager

- Allowing an MQ client to connect to the next available queue manager in a list of queue managers:
- ConnectionNameList



ConnectionNameList (1)

- Both non-reconnectable clients (such as amqsputc) and reconnectable (amqsphac) can exploit the feature: ConnectionNameList

- Let's assume that we have 2 separate queue managers.
- Each queue manager has the local queue Q1.
- Every time that the application starts it should connect to either one and put a message.
- If one of the queue managers is down, then the application should try the other queue manager

ConnectionNameList (2)

- A ConnectionNameList specifies a list of 1 or more pairs of Host(Port) separated by commas.
- This example is for 2 separate queue managers:

- Windows:
- set MQSERVER=SYSTEM.DEF.SVRCONN/TCP/HOST1(1414),HOST2(1415)
- Unix (need to enclose pairs within single quotes)
- export MQSERVER=SYSTEM.DEF.SVRCONN/TCP/'HOST1(1414),HOST2(1415)'

- For the mqclient.ini, inside the Channels stanza:
- ServerConnectionParms=SYSTEM.DEF.SVRCONN/TCP/HOST1(1414),HOST2(1415)

ConnectionNameList (3) - amqsputc

- Using non-reconnectable apps: only at connect time (startup):
- QM_75 is running in HOST1(1414)
- QM_MIG in HOST2(1415)
- Both are up and running.
- We do NOT specify the queue manager:

- C:\> amqsputc Q1
- Sample AMQSPUT0 start
- target queue is Q1
- message-1

ConnectionNameList (4) - amqsputc

- Question:
- Which queue manager gets the message?

- Answer:
- The order in the ConnectionNameList is important:
- HOST1(1414) [QM_75] will be tried first, because it is the first pair in the list.
- If client cannot connect, then it will try the next
- HOST2(1415) [QM_MIG]

- Here, the message is received by QM_75.

ConnectionNameList (5)

- Let's stop QM_75: endmqm -i QM_75

- Then try the client again:
- C:\> amqsputc Q1
- Sample AMQSPUT0 start
- target queue is Q1
- Message-2

- Because QM_75 is down, the client will try to connect to QM_MIG and it succeeds.
- Thus, the Message-2 is received by QM_MIG.

ConnectionNameList – amqsphac

- Let's use a reconnectable client: amqsphac

- C:\> amqsphac Q1
- Sample AMQSPHAC start
- target queue is Q1
- message <Message 1> => Received by QM_75
- message <Message 2> => Received by QM_75
- => endmqm -ir QM_75
- 11:47:52 : EVENT : Connection Reconnecting (Delay: 227ms)
- 11:47:53 : EVENT : Connection Reconnected
- => Connected now to QM_MIG
- message <Message 3> => Received by QM_MIG
- message <Message 4> => Received by QM_MIG

ConnectionNameList – Multi-Instance (1)

- A variation is to use a Multi-Instance queue manager.
- In our example is ANGELMI

- One instance is in host aemaix1(1451), is the Active and
- another instance is in host aemaix2(1451), is the Standby

- Each were started by: strmqm -x ANGELMI
- (the -x indicates to run in multi-instance mode)

- Let's define the MQSERVER:
- C:\> set MQSERVER=SYSTEM.DEF.SVRCONN/TCP/ aemaix1.x.com(1451),aemaix2.x.com(1451)

ConnectionNameList – Multi-Instance (2)

- When using amqsphac, the client will connect to the Active instance, and will be able to put messages.
- The Active in aemaix1 is ended with a switch over:
  endmqm -is ANGELMI
- Note: The -s flag indicates to switch over AND to alert the reconnectable cliens to try to reconnect.
- This will make the Standby in aemaix2 to become Active.
- The client amqsphac will connect to the new Active in aemaix2.

Notes: multi-instance - example

notes

- HOST1: dspmq -x -m ANGELMI
- QMNAME(ANGELMI) STATUS(Running)
- INSTANCE(aemaix1) MODE(Active) INSTANCE(aemaix2) MODE(Standby)
- .
- HOST2: dspmq -x -m ANGELMI
- QMNAME(ANGELMI) STATUS(Running as standby)
- INSTANCE(aemaix1) MODE(Active) INSTANCE(aemaix2) MODE(Standby)
- .
- CLIENT HOST:
- C:\>>amqsphac Q1
- Sample AMQSPHAC start
- target queue is Q1
- message <Message 1>
- .
- HOST1: endmqm -is ANGELMI

- HOST2: dspmq -x -m ANGELMI
- QMNAME(ANGELMI) STATUS(Running) INSTANCE(aemaix2) MODE(Active)

- CLIENT HOST:
- 14:05:33 : EVENT : Connection Reconnecting (Delay: 75ms)
- 14:05:38 : EVENT : Connection Reconnecting (Delay: 52ms)
- 14:05:40 : EVENT : Connection Reconnected
- message <Message 2>

Client Channel Definition Table (CCDT)

- The last mechanism is:
- Client Channel Definition Table (CCDT)
- Need to unset MQSERVER and mqclient.ini
- (Common pitfall) Thinking that the CCDT is being used, but actually MQSERVER or mqclient.ini is being used first.

- For more details see:
- http://www.ibm.com/support/docview.wss?uid=swg27024109
- WSTE: Using a Client Channel Definition Table (CCDT) in MQ V7 for Queue Manager Groups

Adding channels to a CCDT

- A CCDT file is generated when the queue manager is created.
- To add channels to the CCDT, you need to define the following:
- 1) A Server Connection Channel, such as:
- DEFINE CHANNEL(ANGELMI) + CHLTYPE(SVRCONN) TRPTYPE(TCP)
-

Adding channels to a CCDT - 2

- 2) Define a corresponding Client Connection Channel with the SAME name as the Server Connection Channel:
- DEFINE CHANNEL(ANGELMI) + CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
- CONNAME('aemaix1.*.com(1451),aemaix2.*.com(1421)') +
- QMNAME(ANGELMI)
- 3) Copy as binary the AMQCLCHL.TAB file into the client host.
- 4) Setup the variables as mentioned in next page
- 5) Perform the test in page 47

Notes: Environment variables for CCDT

notes

Setup of environment variables for using a CCDT file

Unix:

```
unset MQSERVER
unset MQCLNTCF
```

Specify the location of the CCDT file. The default is:

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

Note: the following is only if the name is not the default: AMQCLCHL.TAB

```
export MQCHLTAB=FILENAME
```

Windows:

```
set MQSERVER=
set MQCLNTCF=
```

Specify the location of the CCDT file. The default is:

```
set MQCHLLIB=C:\var\mqm\Qmgrs\<QMgrName>\@ipcc
```

Note: the following is only if the name is not the default: AMQCLCHL.TAB

```
set MQCHLTAB=FILENAME
```

CCDT – detailed techdocs

- http://www.ibm.com/support/docview.wss?uid=swg27020848
- Using a Client Channel Definition Table (CCDT) in WebSphere MQ V7 for Queue Manager Groups

- http://www.ibm.com/support/docview.wss?uid=swg27020862
- Using a CCDT file to connect to multiple WebSphere MQ queue managers using JMS

Notes: Using MQ CCDT from WAS

notes

- The following WSTE has a recorded audio file in MP3 format and written material in a PDF file and offers a generat overview:
- Using Custom Property and CCDT File to Connect to WebSphere MQ Multi-Instance Queue Managers from WebSphere Application Server V7
- http://www-01.ibm.com/support/docview.wss?uid=swg27020901
-
- The following technical documents provide more specific steps:

- Using WebSphere MQ automatic client reconnection with the WebSphere MQ classes for JMS
- http://www.ibm.com/support/docview.wss?uid=swg21508357

- Using custom property connectionNameList to connect to MQ multi-instance queue managers from WAS http://www.ibm.com/support/docview.wss?rs=171&uid=swg27020700

- Using a CCDT file to connect to WebSphere MQ multi-instance queue managers from WAS V7
- http://www.ibm.com/support/docview.wss?rs=171&uid=swg27020701

- Using CCDT in an Activation Specification for a Queue Manager Group of separate queue managers in WAS V7
- http://www.ibm.com/support/docview.wss?rs=171&uid=swg27035714

- Using CCDT in Connection Factory for a Queue Manager Group of separate queue managers in WAS V7
- http://www.ibm.com/support/docview.wss?rs=171&uid=swg27035452

XA and CCDT

- http://www-01.ibm.com/support/docview.wss?uid=swg21631879
- Configuring WebSphere Application Server to connect via a CCDT to a WMQ queue manager for Global Transaction XA capability.

- Not Supported:
- - CCDT to connect to multiple different queue managers
- - CCDT to connect to multiple multi-instance q managers
- Supported:
- - CCDT to connect to single queue manager
- - CCDT to connect to a single multi-instance queue manager

Additional WebSphere Product Resources

- Learn about upcoming WebSphere Support Technical Exchange webcasts, and access previously recorded presentations at:
  http://www.ibm.com/software/websphere/support/supp_tech.html
- Discover the latest trends in WebSphere Technology and implementation, participate in technically-focused briefings, webcasts and podcasts at:
  http://www.ibm.com/developerworks/websphere/community/
- Join the Global WebSphere Community:
  http://www.websphereusergroup.org
- Access key product show-me demos and tutorials by visiting IBM® Education Assistant:
  http://www.ibm.com/software/info/education/assistant
- View a webcast replay with step-by-step instructions for using the Service Request (SR) tool for submitting problems electronically:
  http://www.ibm.com/software/websphere/support/d2w.html
- Sign up to receive weekly technical My Notifications emails:
  http://www.ibm.com/software/support/einfo.html

Connect with us!

1. **Get notified on upcoming webcasts**
   Send an e-mail to wsehelp@us.ibm.com with subject line “wste subscribe” to get a list of mailing lists and to subscribe
2. **Tell us what you want to learn**
   Send us suggestions for future topics or improvements about our webcasts to wsehelp@us.ibm.com
3. **Be connected!**
   Connect with us on Facebook
   Connect with us on Twitter

Questions and Answers