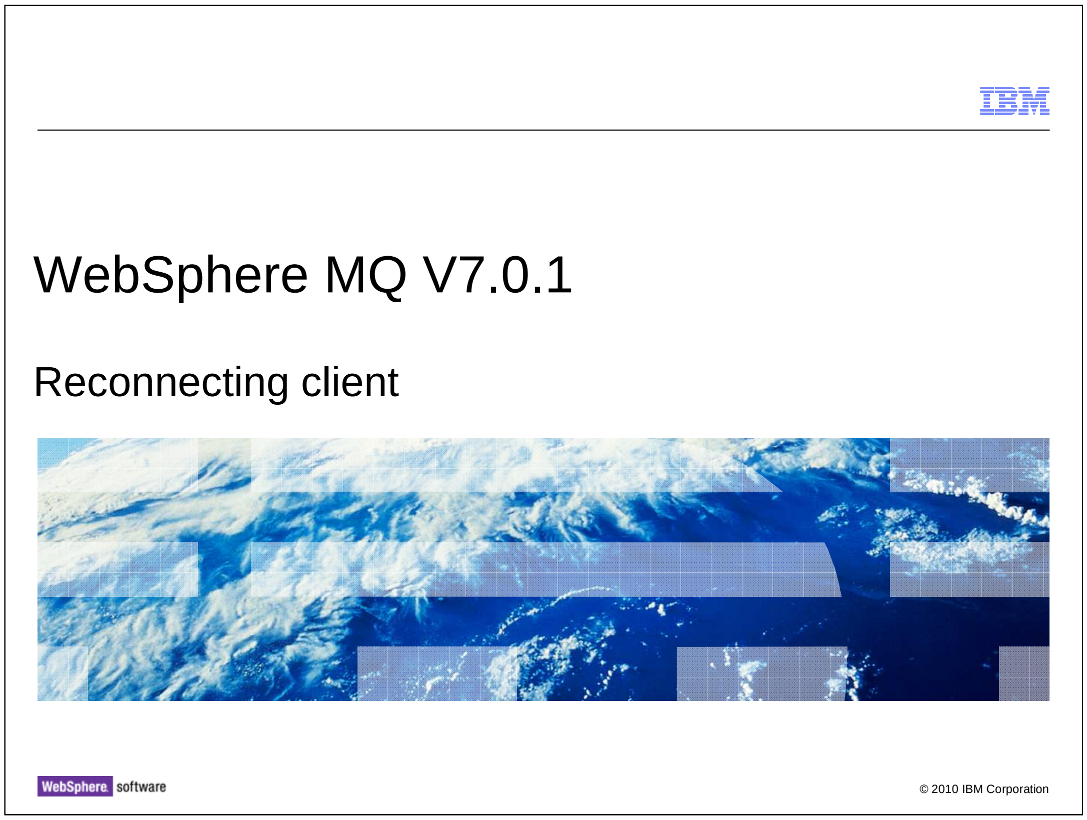
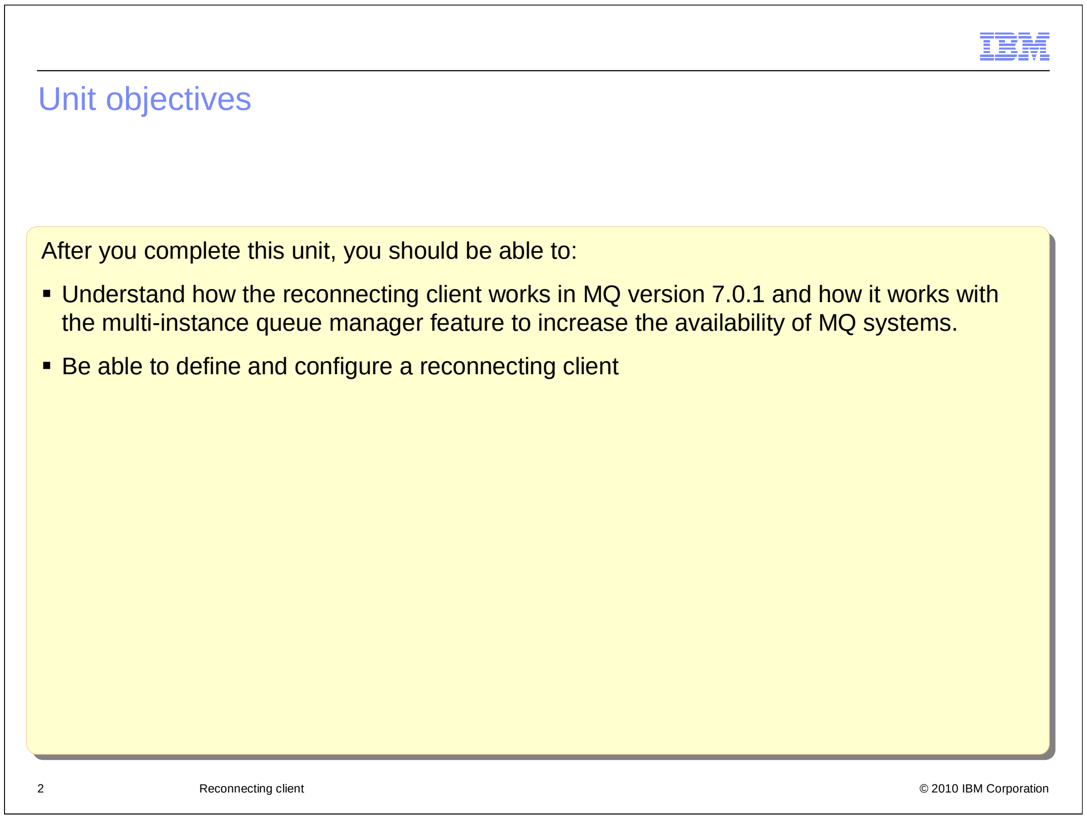


WebSphere MQ V7.0.1

Reconnecting client

This unit explains the multi-instance queue manger feature introduced in WebSphere® MQ version 7.0.1.

This unit assumes a reasonable understanding of how WebSphere MQ works.



Unit objectives

After you complete this unit, you should be able to:

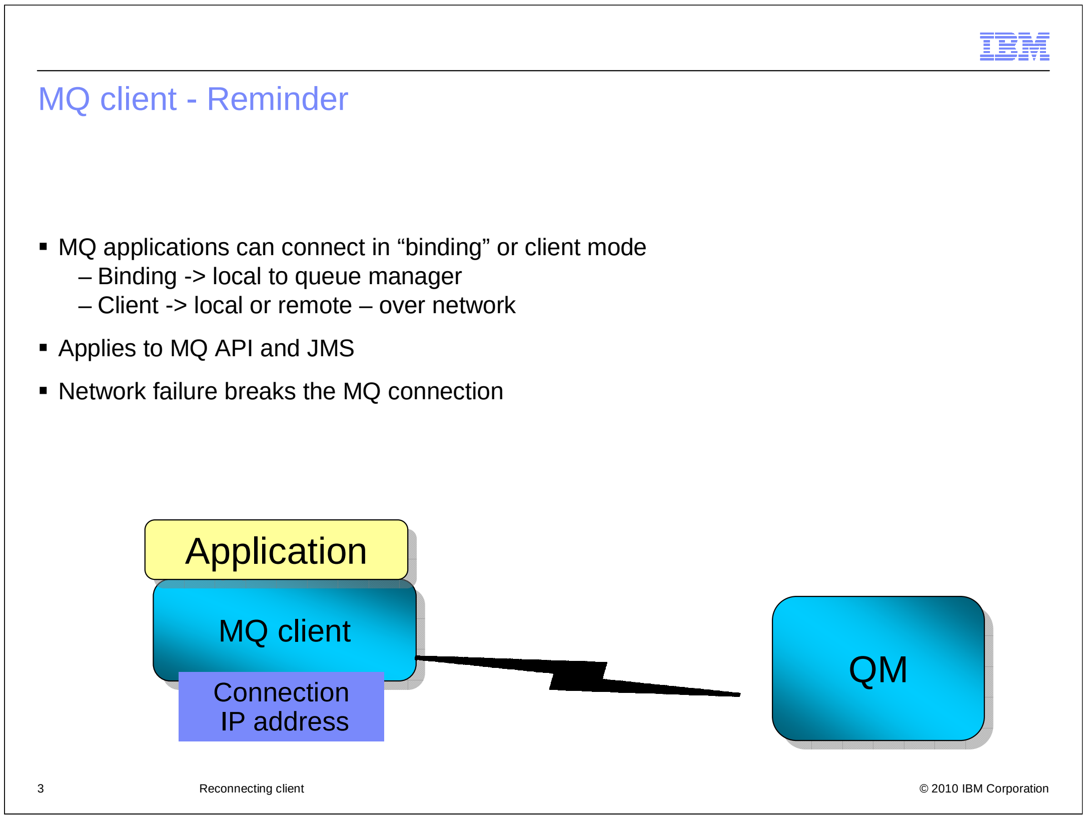
- Understand how the reconnecting client works in MQ version 7.0.1 and how it works with the multi-instance queue manager feature to increase the availability of MQ systems.
- Be able to define and configure a reconnecting client

2 Reconnecting client © 2010 IBM Corporation

After you complete this unit you should have some understanding of how existing applications using the queued publish / subscribe mechanism is migrated to MQ version 7.

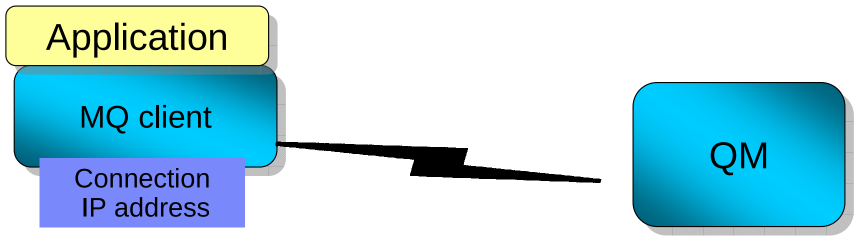
You will also be aware of some of the issues that need to be considered before migration.

This unit does not attempt to cover the full range of syntax and options available, for which you should refer to the product information center.



MQ client - Reminder

- MQ applications can connect in “binding” or client mode
  - Binding -> local to queue manager
  - Client -> local or remote – over network
- Applies to MQ API and JMS
- Network failure breaks the MQ connection



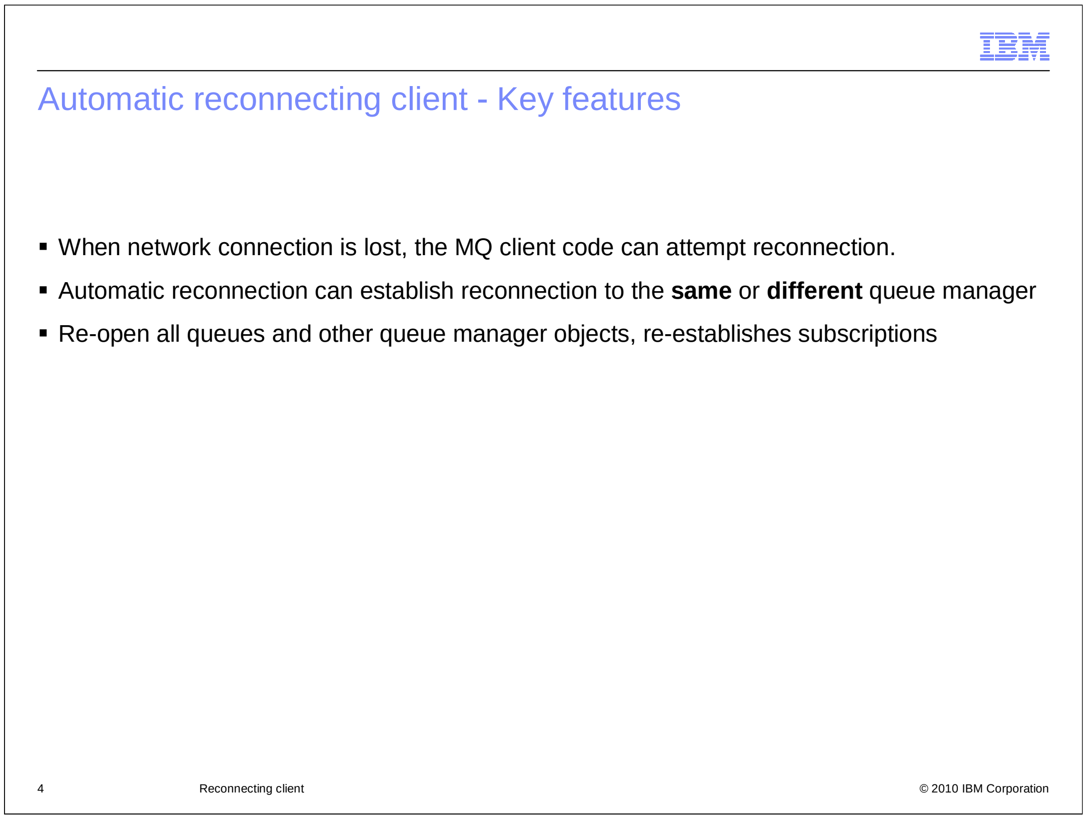
3 Reconnecting client © 2010 IBM Corporation

WebSphere MQ applications need to be connected to a queue manager in order to function. Basically two methods of connecting are supported:

Binding mode, only possible where the application is running on the same machine as the queue manager, where the application issues IPA calls fairly directly with the queue manager. And client mode where the application makes API calls to an MQ client which conveys the commands to a queue manager over a communications link, typically TCP/IP.

This choice applies equally to applications written to the MQ API (the MQI) and to applications using abstractions like the Java™ Messaging Service (JMS).

One issue with using client connections is that the MQ session is lost if the network fails or if the queue manager fails. In this case the application program is notified and must terminate or run other user written code to recover or circumvent the issue.



Automatic reconnecting client - Key features

- When network connection is lost, the MQ client code can attempt reconnection.
- Automatic reconnection can establish reconnection to the **same** or **different** queue manager
- Re-open all queues and other queue manager objects, re-establishes subscriptions

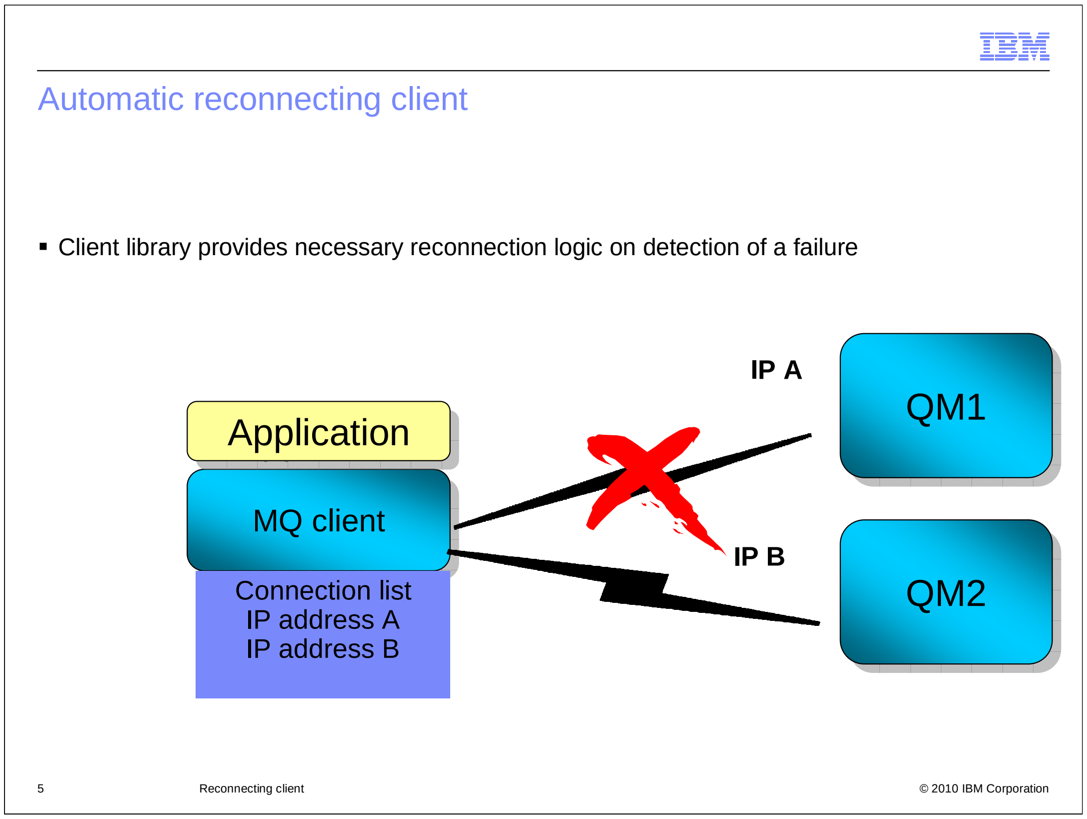
4 Reconnecting client © 2010 IBM Corporation

The new reconnecting client feature overcomes some of these problems by running reconnect logic in the event of an unexpected loss of the network connection. This feature is only available in the TCP/IP client.

The client configuration has been extended to allow a list of possible IP address and port values to connect to. The reconnection attempt can be to ANY queue manager, or to only the same queue manager, but note that this may mean another instance of the same queue manager. This supports the case where a client is connected to the active instance of a queue manager; this instance fails; the standby instance starts up on a different address and the previously connected clients can automatically reconnect to this instance.

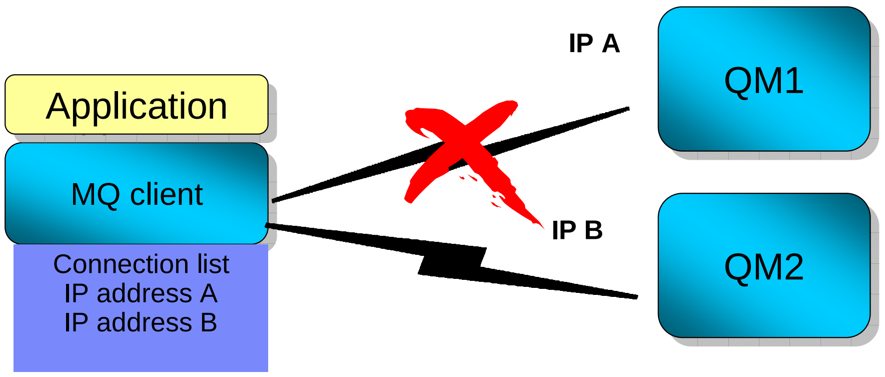
After reconnecting the MQ client code attempts to restore the client application to the same state it was in when the connection failed by opening all the QUEUES, TOPICS and so on that were in use at the point of failure.

When all this work is completed the application program continues.



Automatic reconnecting client

- Client library provides necessary reconnection logic on detection of a failure

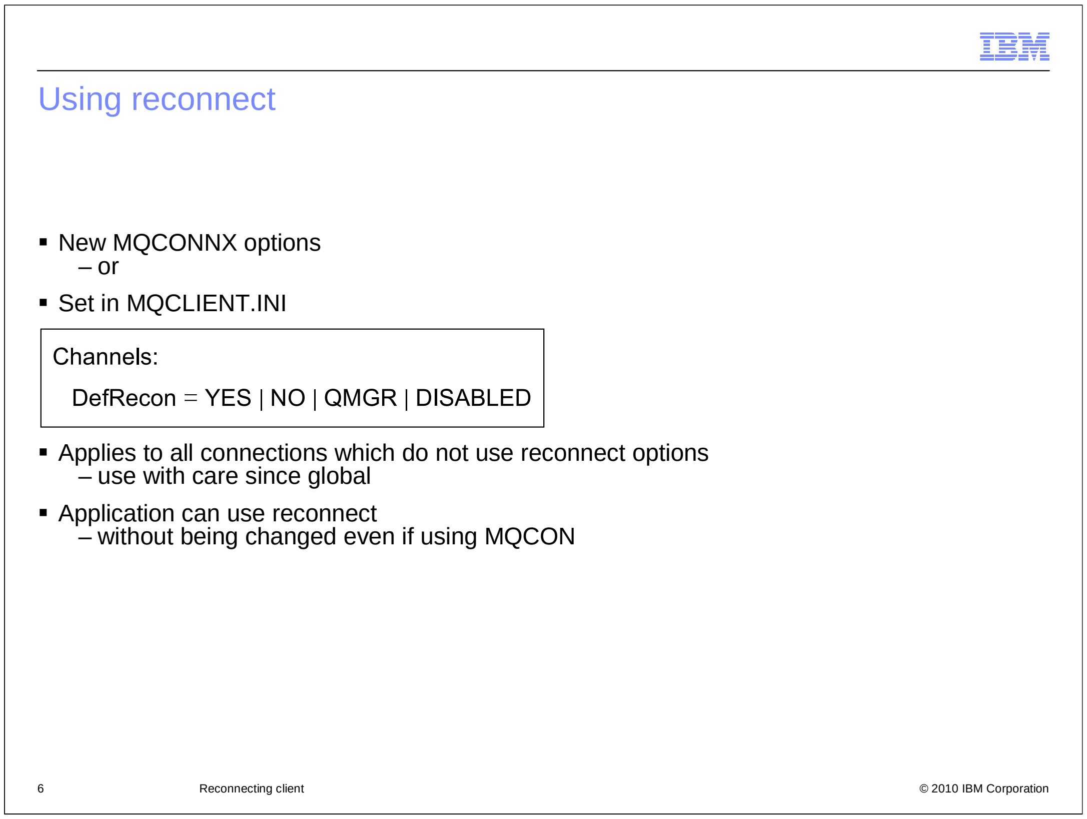


5 Reconnecting client © 2010 IBM Corporation

With the reconnecting client feature a switch can be made from one queue manager on one IP address to another (or another instance of the same) queue manager on another IP address.

For this the MQ client configuration date needs a list of the possible locations the client may need to connect to.

When the MQ client code reconnects it will reopen the queues and topics in use and renew subscriptions but you should be aware that some states are not restored. For example cursor position if browsing a queue.



Using reconnect

- New MQCONNX options
  – or
- Set in MQCLIENT.INI

```
Channels:
  DefRecon = YES | NO | QMGR | DISABLED
```

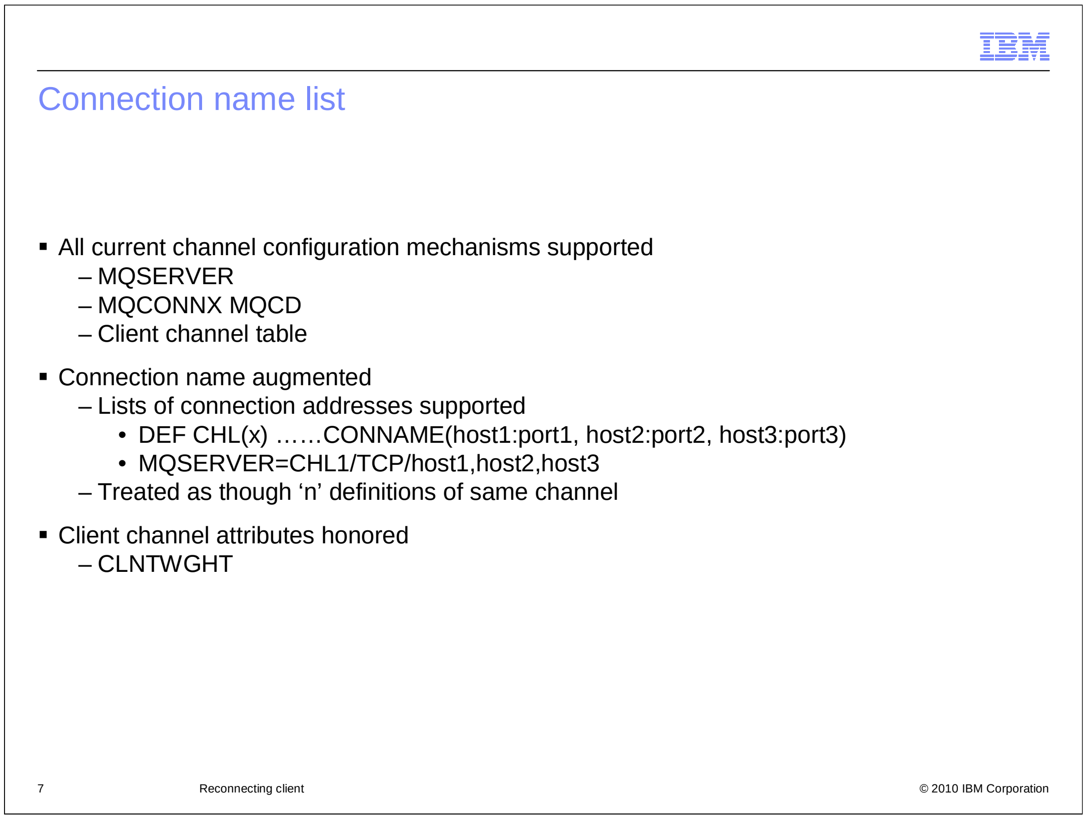
- Applies to all connections which do not use reconnect options
  – use with care since global
- Application can use reconnect
  – without being changed even if using MQCON

6 Reconnecting client © 2010 IBM Corporation

Reconnect can be enabled either by using new options in the MQCONNX call or by adding the information to the “channels” stanza in the MQCLIENT.INI file.

The DefRecon (Default reconnection) value can be: YES – reconnect is enabled unless an MQCONNX call indicates do not use, NO – reconnect is disabled unless an MQCONNX call indicates it should be used, QMGR – like yes but only connect to the same queue manager (possibly a different instance), DISABLED – regardless of MQCONNX options, do not enable the reconnect feature.

Note that the option applies to all connections from this client, and this means that existing programs can use the reconnect feature without any modification or recompiling.


Connection name list

- All current channel configuration mechanisms supported
  - MQSERVER
  - MQCONNX MQCD
  - Client channel table
- Connection name augmented
  - Lists of connection addresses supported
    - DEF CHL(x) ……CONNAME(host1:port1, host2:port2, host3:port3)
    - MQSERVER=CHL1/TCP/host1,host2,host3
  - Treated as though ‘n’ definitions of same channel
- Client channel attributes honored
  - CLNTWGHT

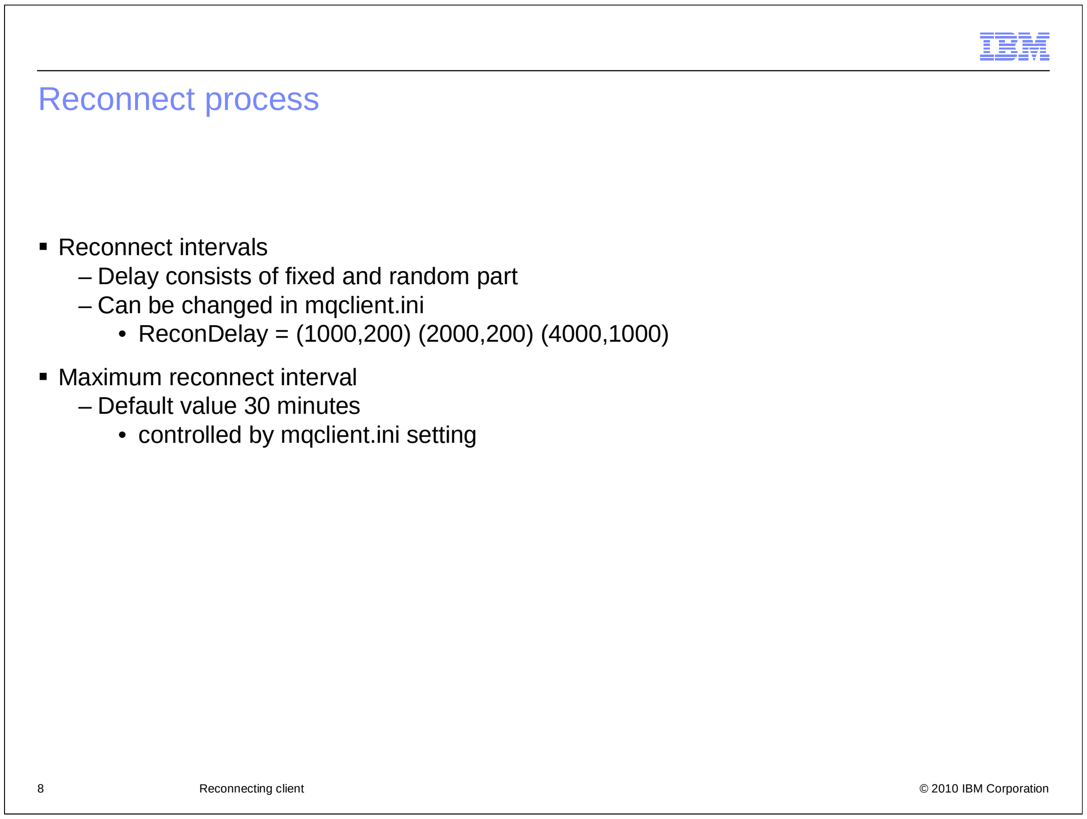
7 Reconnecting client © 2010 IBM Corporation

As mentioned earlier one requirement to make the reconnecting client function is a list of possible connection addresses.

These lists can be specified in all the ways a current connection can be specified. he MQSERVER environment variable, the MQCONNX MQCD option and the Client Channel Definition Table.

Effectively where channels had a connection name that identified a single IP address as the network address a list of such IP addresses is now allowed. Connection to any of these addresses would be valid for this channel.

The client weight attribute is still honored. This is used when a client definition table is used to allow connection to a range of possible queue managers to determine the ratio of connections to specific hosts. The weights will still apply to the channel which could lead to connecting to any of the connection names.



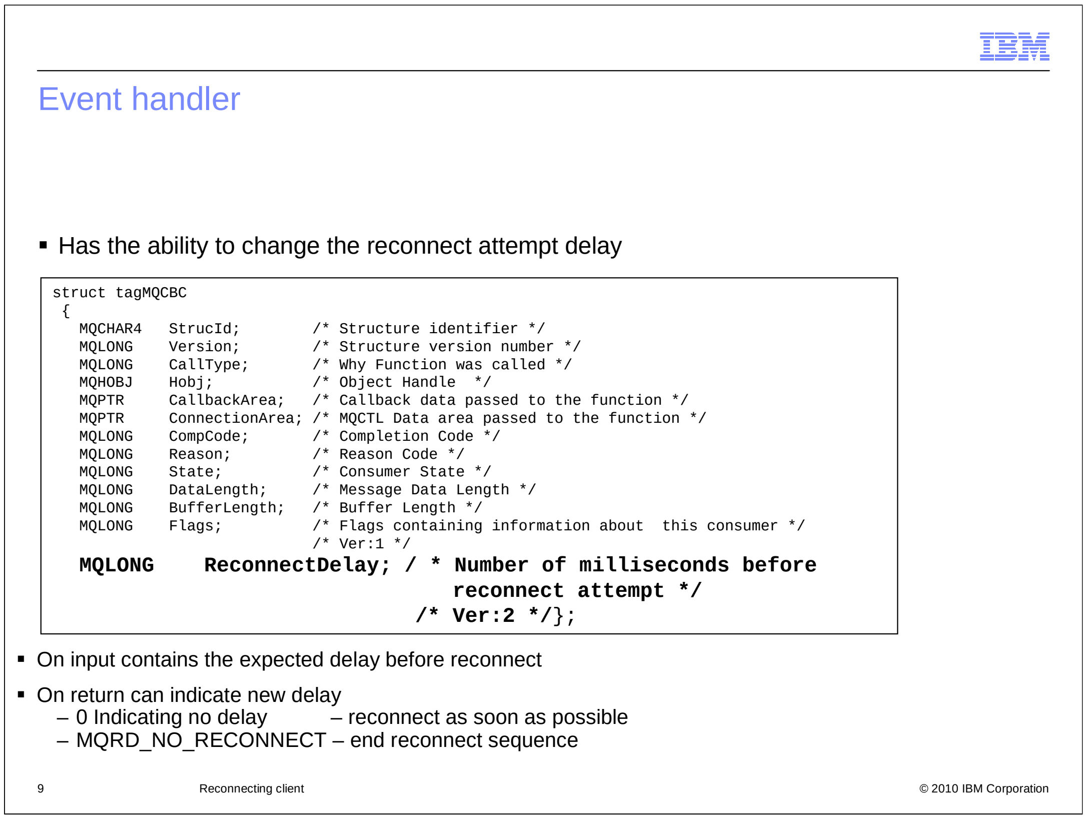
Reconnect process

- Reconnect intervals
  - – Delay consists of fixed and random part
  - – Can be changed in mqclient.ini
    - • ReconDelay = (1000,200) (2000,200) (4000,1000)
- Maximum reconnect interval
  - – Default value 30 minutes
    - • controlled by mqclient.ini setting

When the connection is lost and the reconnection process begins the reconnection process is tried repeatedly until reconnection is made or until the maximum reconnect interval occurs.

In order to stop all the clients attached to a failed queue manager reconnecting at once the reconnects are delayed by intervals which are part fixed and part random.

The default values should be appropriate in most cases, but can be overridden in the MQCLIENT.INI. The example shown would indicate an initial delay of 1 second plus a random element up to 200 milliseconds, the next delay would be 2 seconds plus a random interval up to 200 milliseconds, all subsequent delays would be 4 seconds plus a random delay up to 1000 milliseconds. Reconnect would then be attempted at this interval for the max interval (default 30 seconds), after this time the connection failure is reported to the application.



Event handler

- Has the ability to change the reconnect attempt delay

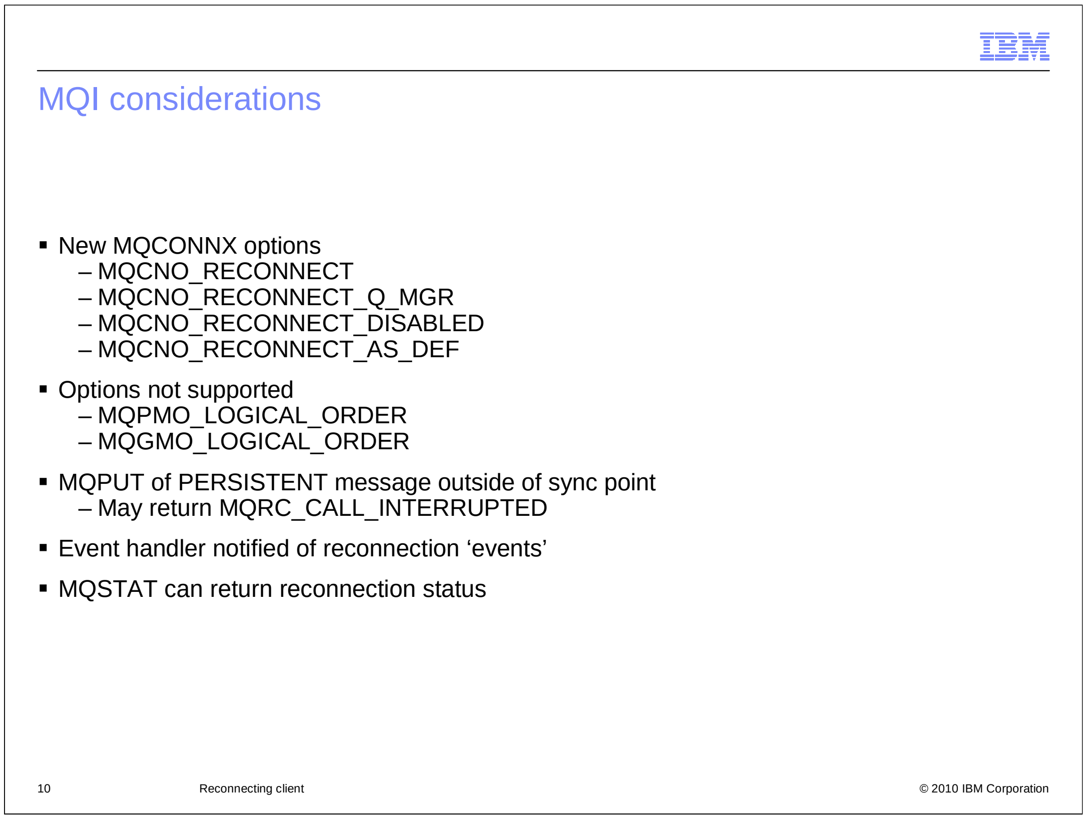
```
struct tagMQCBC
 {
   MQCHAR4   StrucId;        /* Structure identifier */
   MQLONG    Version;        /* Structure version number */
   MQLONG    CallType;       /* Why Function was called */
   MQHOBJ    Hobj;           /* Object Handle  */
   MQPTR     CallbackArea;   /* Callback data passed to the function */
   MQPTR     ConnectionArea; /* MQCTL Data area passed to the function */
   MQLONG    CompCode;       /* Completion Code */
   MQLONG    Reason;         /* Reason Code */
   MQLONG    State;          /* Consumer State */
   MQLONG    DataLength;     /* Message Data Length */
   MQLONG    BufferLength;   /* Buffer Length */
   MQLONG    Flags;          /* Flags containing information about  this consumer */
                             /* Ver:1 */
   MQLONG    ReconnectDelay; / * Number of milliseconds before
                                 reconnect attempt */
                              /* Ver:2 */};
```

- On input contains the expected delay before reconnect
- On return can indicate new delay
  – 0 Indicating no delay – reconnect as soon as possible
  – MQRD_NO_RECONNECT – end reconnect sequence

9 Reconnecting client © 2010 IBM Corporation

When using the client reconnect feature application programs would normally be unaware of reconnects that occur. However if an application needs to be informed of reconnection activities it is possible to use an Event Handler. Event handlers were introduced in version 7.0 along with the asynchronous message consume feature.

If an event handler is setup it will be invoked during reconnect activity and has the opportunity to change the delay interval by setting RecconnectDelay.



MQI considerations

- New MQCONNX options
  - MQCNO_RECONNECT
  - MQCNO_RECONNECT_Q_MGR
  - MQCNO_RECONNECT_DISABLED
  - MQCNO_RECONNECT_AS_DEF
- Options not supported
  - MQPMO_LOGICAL_ORDER
  - MQGMO_LOGICAL_ORDER
- MQPUT of PERSISTENT message outside of sync point
  - May return MQRC_CALL_INTERRUPTED
- Event handler notified of reconnection ‘events’
- MQSTAT can return reconnection status

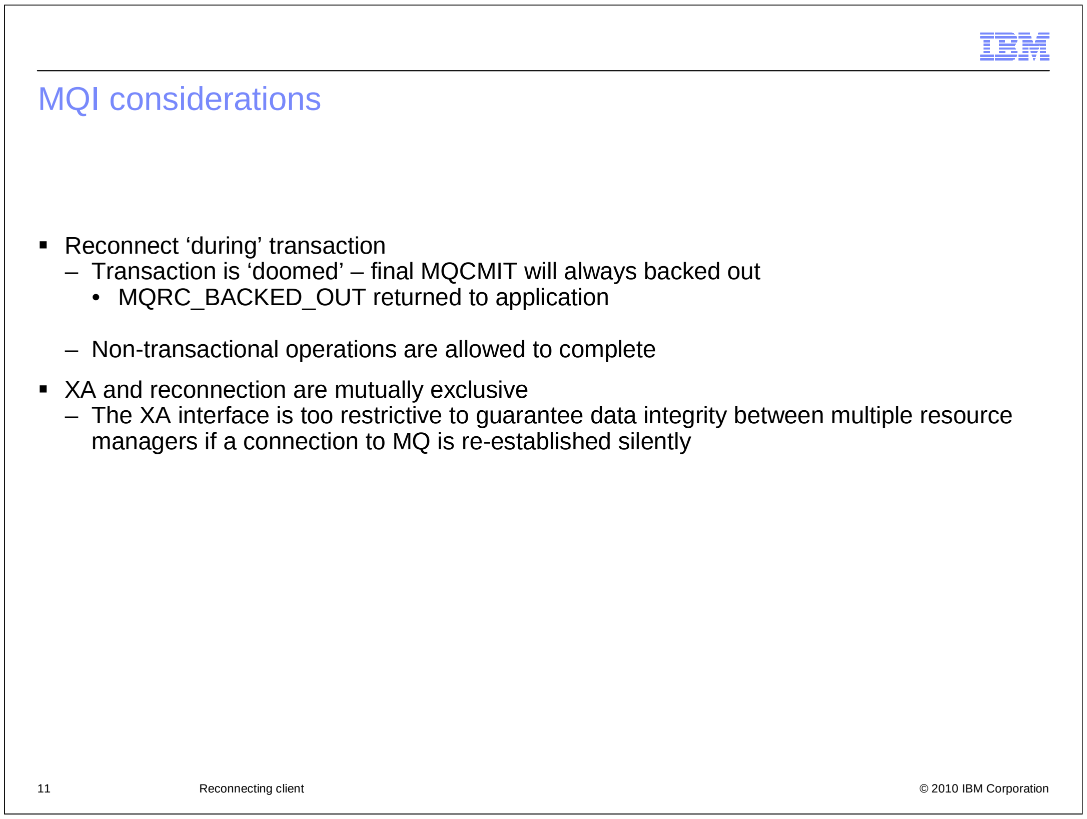
10 Reconnecting client © 2010 IBM Corporation

Now take a look at the implications for the WebSphere MQ API.

The makor change is an additional set of connection options for the MQCONNX call. MQCNO_RECONNECT means that reconnecting client is preferred for this connection. The MQCNO_RECONNECT_Q_MGR means the same but reconnect is only allowed the same queue manager, including other instances of a queue manager. MQCNO_RECONNECT_DISABLED inhibits the use of the reconnecting client. The default value is MQCNO_RECONNECT_AS_DEF, which means that the reconnecting client is used if it is configured for this client, the default value if no specification is made for the client definition is not to use reconnect.

Certain MQGET and PUT options are not compatible with reconnect, full details can be found in the product information center.

New reason code of MQRC_CALL_INTERRUPTED might be returned if the client connection reconnects “within” an MQPUT call and may need coding for. As mentioned earlier the asynchronous event handler can be notified of reconnection events and the MQSTAT call is extended to return reconnection information.



MQI considerations

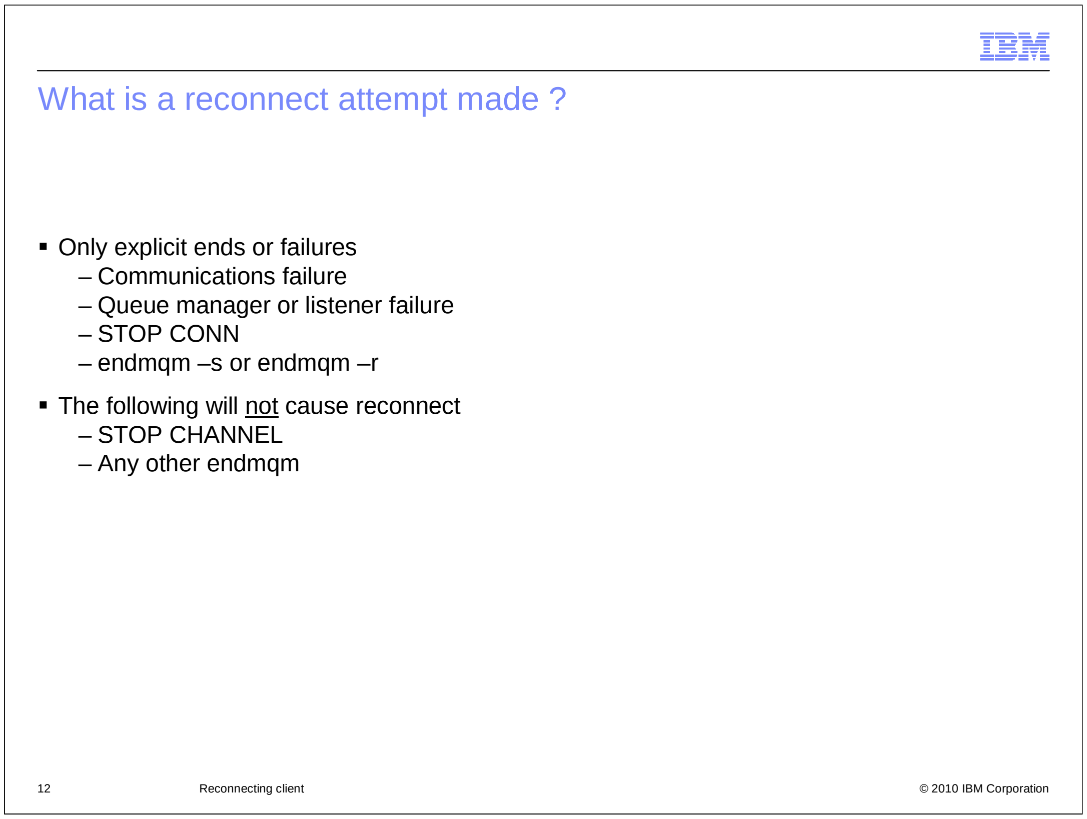
- Reconnect ‘during’ transaction
  - Transaction is ‘doomed’ – final MQCMIT will always backed out
    - MQRC_BACKED_OUT returned to application
  - Non-transactional operations are allowed to complete
- XA and reconnection are mutually exclusive
  - The XA interface is too restrictive to guarantee data integrity between multiple resource managers if a connection to MQ is re-established silently

11 Reconnecting client © 2010 IBM Corporation

The final considerations relate to activity within a sync point.

The major point is that if a client reconnect occurs within a unit of work, the unit of work is DOOMED. Although work can continue after the reconnect the final commit operation will result in the transaction being backed out.

Also limitation on the XA interface means that XA transactions – such as those coordinated WebSphere Application Server – are incompatible with the reconnecting client.



What is a reconnect attempt made ?

- 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

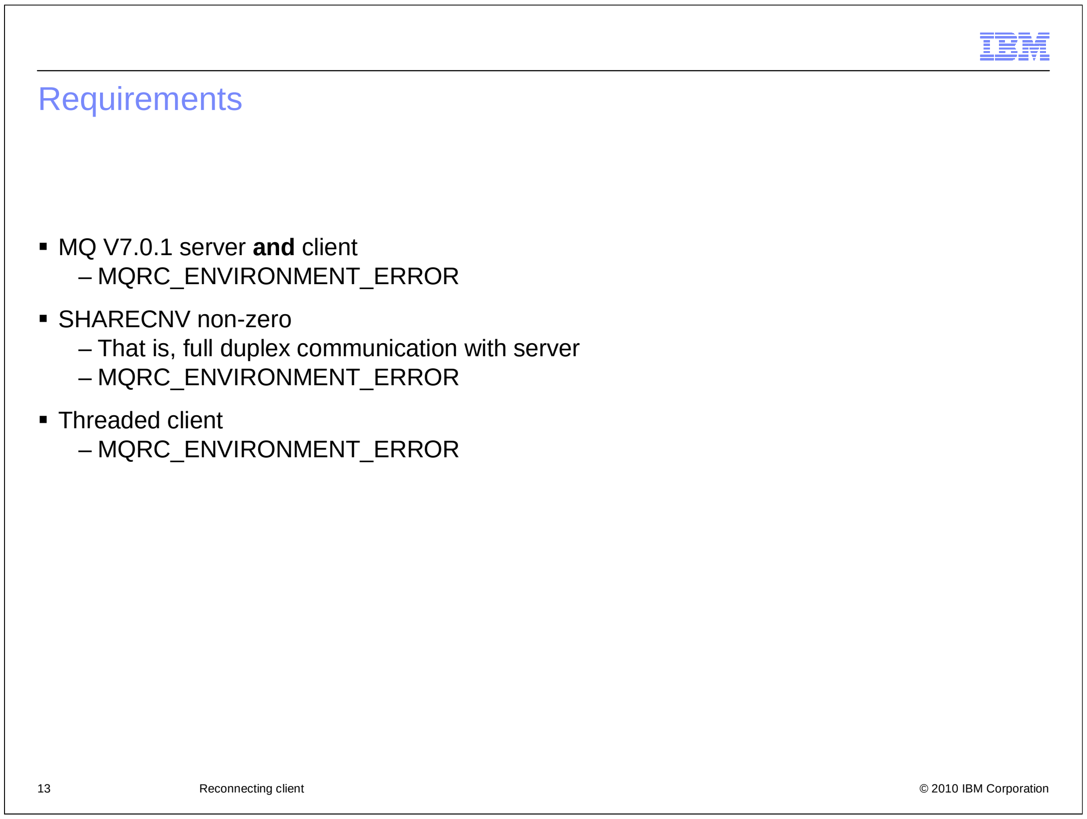
12 Reconnecting client © 2010 IBM Corporation

The question now arises as to under what circumstances the reconnecting client will in fact attempt to reconnect.

These are summarized on this slide.

Failure of the communications network so that the client can no longer reach the queue manager. A failure of a queue manager or listener process, or of course the machine they are running on. A stop connection command. An endmqm command BUT ONLY if it has one of the two new options “-r” explicitly requesting a reconnect, or “-s” which explicitly requests a switch to a standby instance.

Any other endmqm will not cause a reconnect to be attempted, nor will a stop of the channel.



Requirements

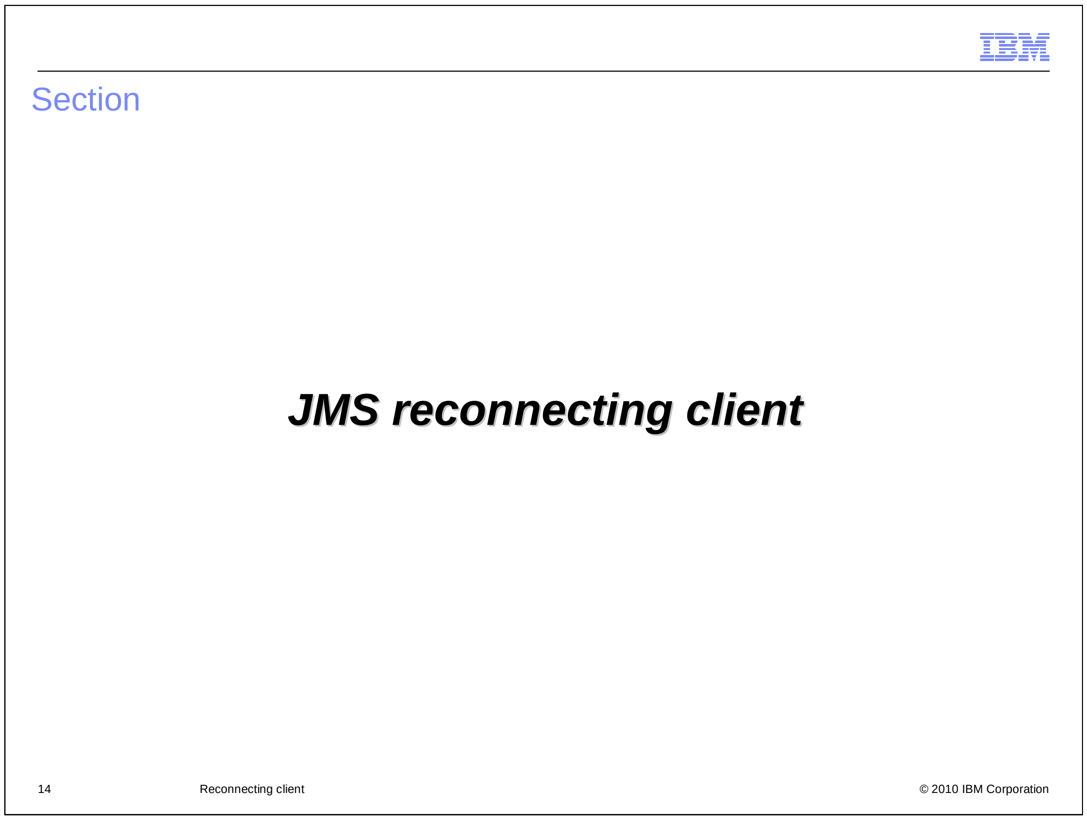
- MQ V7.0.1 server **and** client
  - MQRC_ENVIRONMENT_ERROR
- SHARECNV non-zero
  - That is, full duplex communication with server
  - MQRC_ENVIRONMENT_ERROR
- Threaded client
  - MQRC_ENVIRONMENT_ERROR

13 Reconnecting client © 2010 IBM Corporation

Some remarks about the prerequisite requirements for the reconnecting client.

First and foremost is that both client and server code must be at version 7.0.1 or better. The channel must be running in full duplex mode, this means that the SHARECONV value must be 1 or greater.

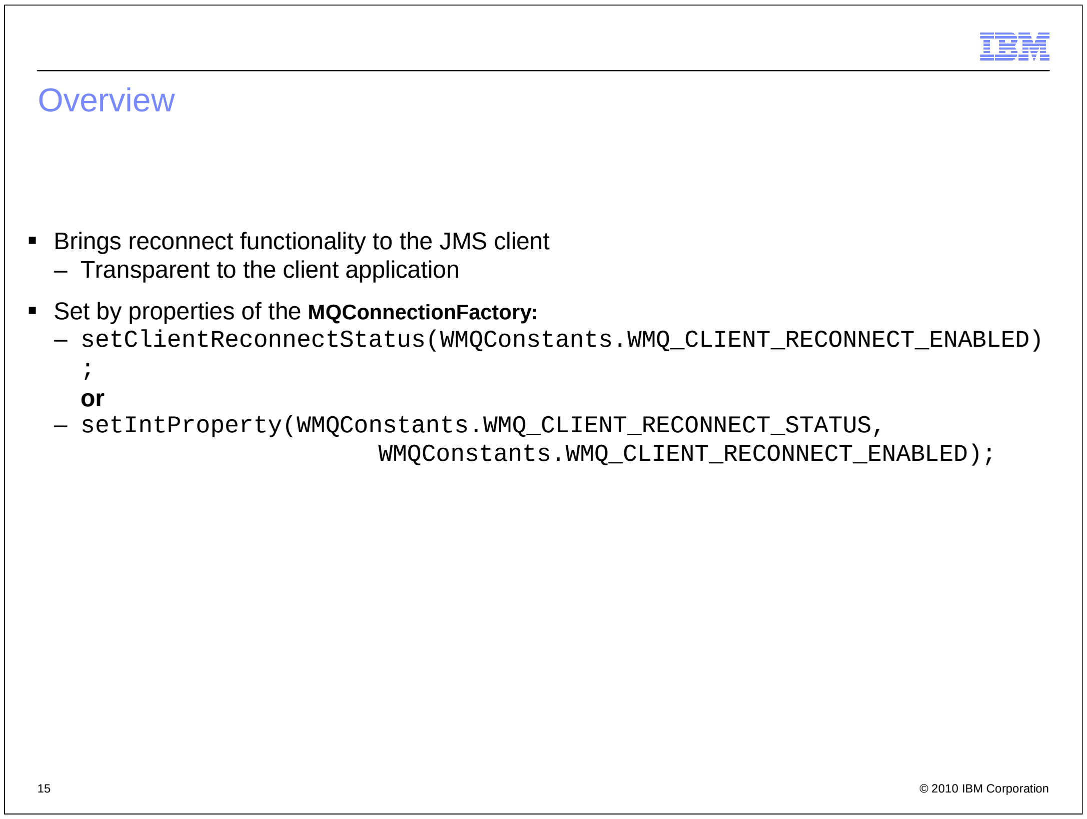
Finally the application must be compiled using the threaded client libraries, not the older unthreaded libraries sometimes used in UNIX® systems.



Section

JMS reconnecting client

So far you have seen the reconnecting client in terms of MQ API applications, but the full benefit is also available to applications using the Java Messaging Service (JMS) APIs.



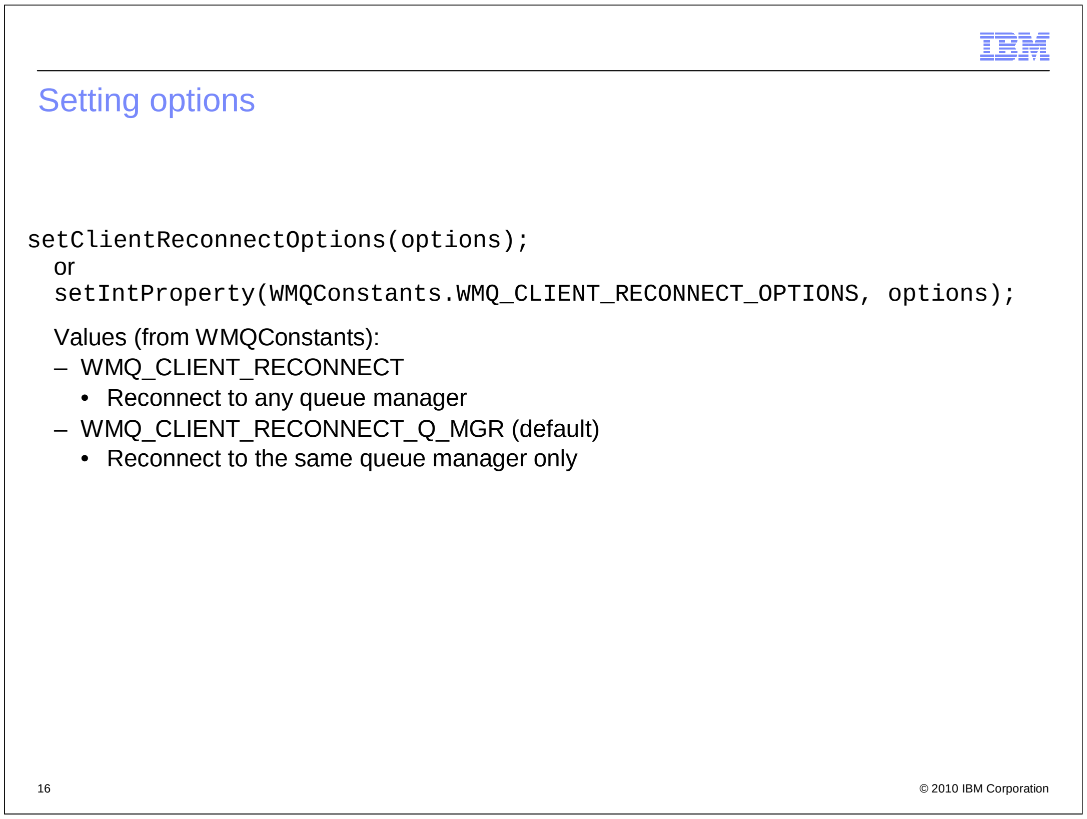
Overview

- Brings reconnect functionality to the JMS client
  – Transparent to the client application
- Set by properties of the **MQConnectionFactory:**
  – `setClientReconnectStatus(WMQConstants.WMQ_CLIENT_RECONNECT_ENABLED);`
  or
  – `setIntProperty(WMQConstants.WMQ_CLIENT_RECONNECT_STATUS,`
  `WMQConstants.WMQ_CLIENT_RECONNECT_ENABLED);`

15

The full functionality of the reconnecting client is made available to JMS applications. It requires no changes to the application programs.

It is enabled by either using the explicit method “setClientReconnectStatus” or by using the more general setIntProperty method to the appropriate value.



Setting options

```
setClientReconnectOptions(options);
```

or

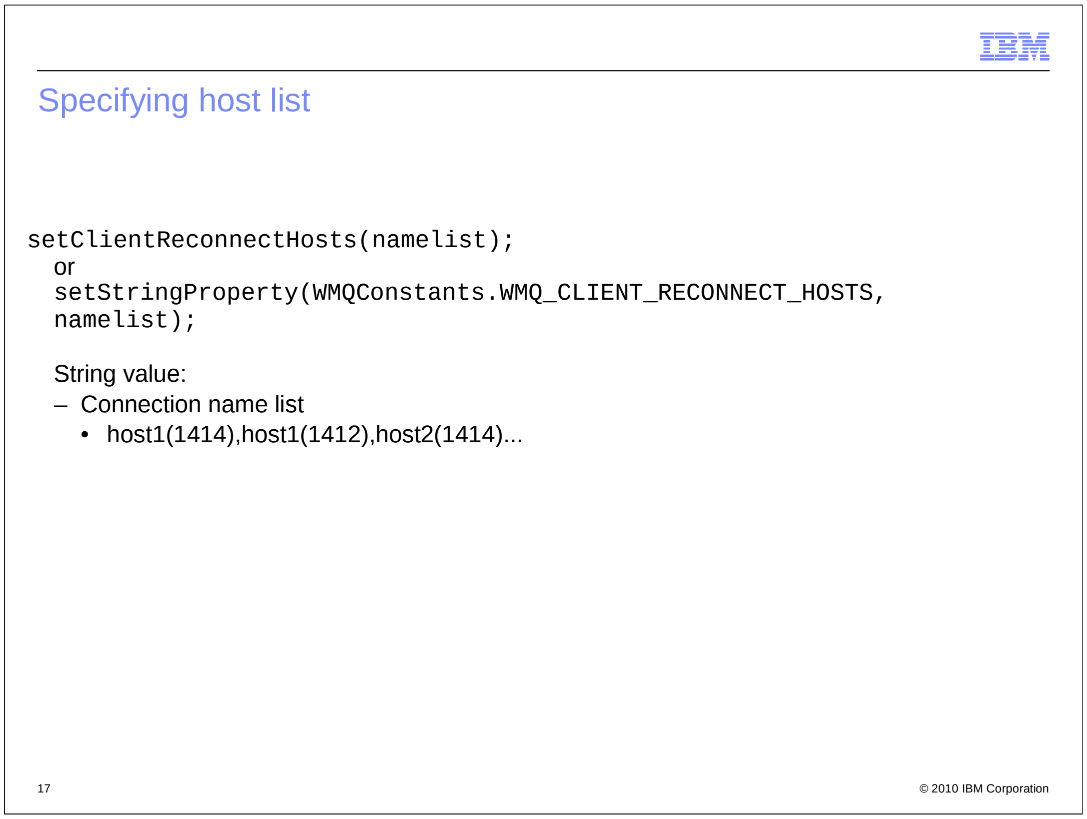
```
setIntProperty(WMQConstants.WMQ_CLIENT_RECONNECT_OPTIONS, options);
```

Values (from WMQConstants):

- WMQ_CLIENT_RECONNECT
  - Reconnect to any queue manager
- WMQ_CLIENT_RECONNECT_Q_MGR (default)
  - Reconnect to the same queue manager only

16

Options can be set using the “setClientReconnectOptions” method, or again the setIntProperty method. In this call it is possible to specify reconnect to only the same queue manager (including another instance) or to any queue manager.



Specifying host list

```
setClientReconnectHosts(namelist);
```

or

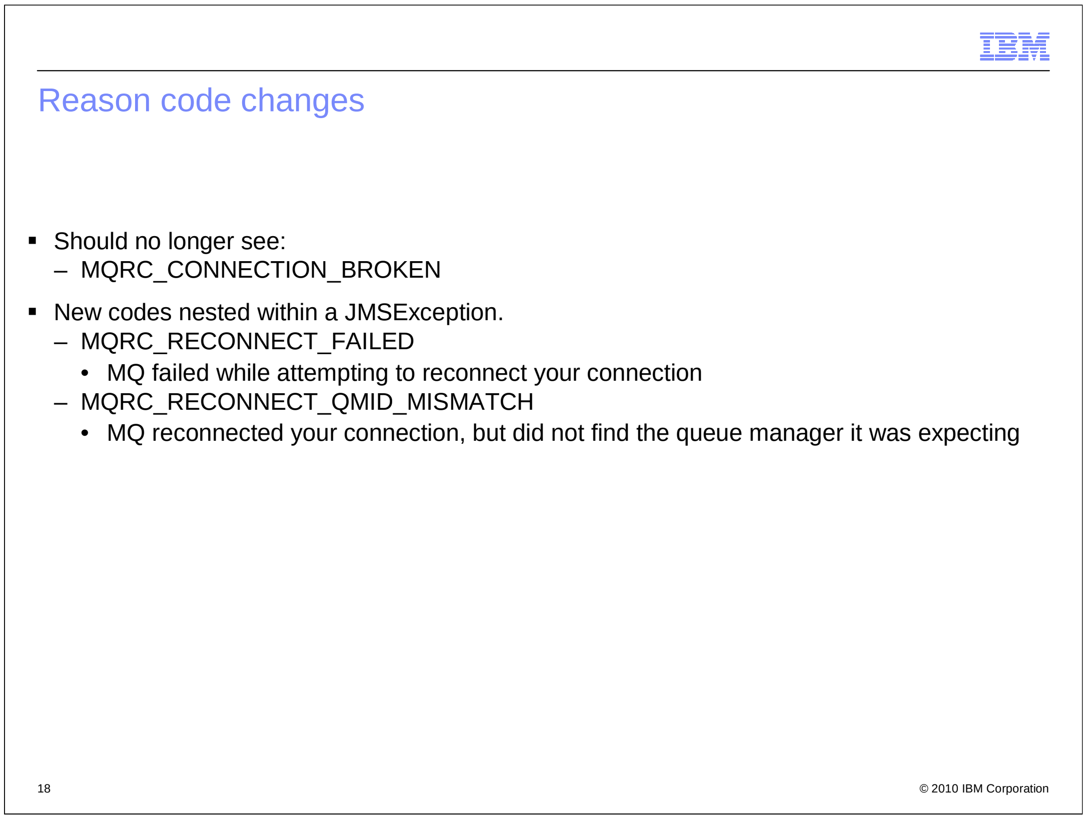
```
setStringProperty(WMQConstants.WMQ_CLIENT_RECONNECT_HOSTS,
namelist);
```

String value:

- Connection name list
  - host1(1414),host1(1412),host2(1414)...

17 © 2010 IBM Corporation

The list of possible hosts to reconnect to is specified using the method" setClientReconnectHosts" to a list of host addresses.

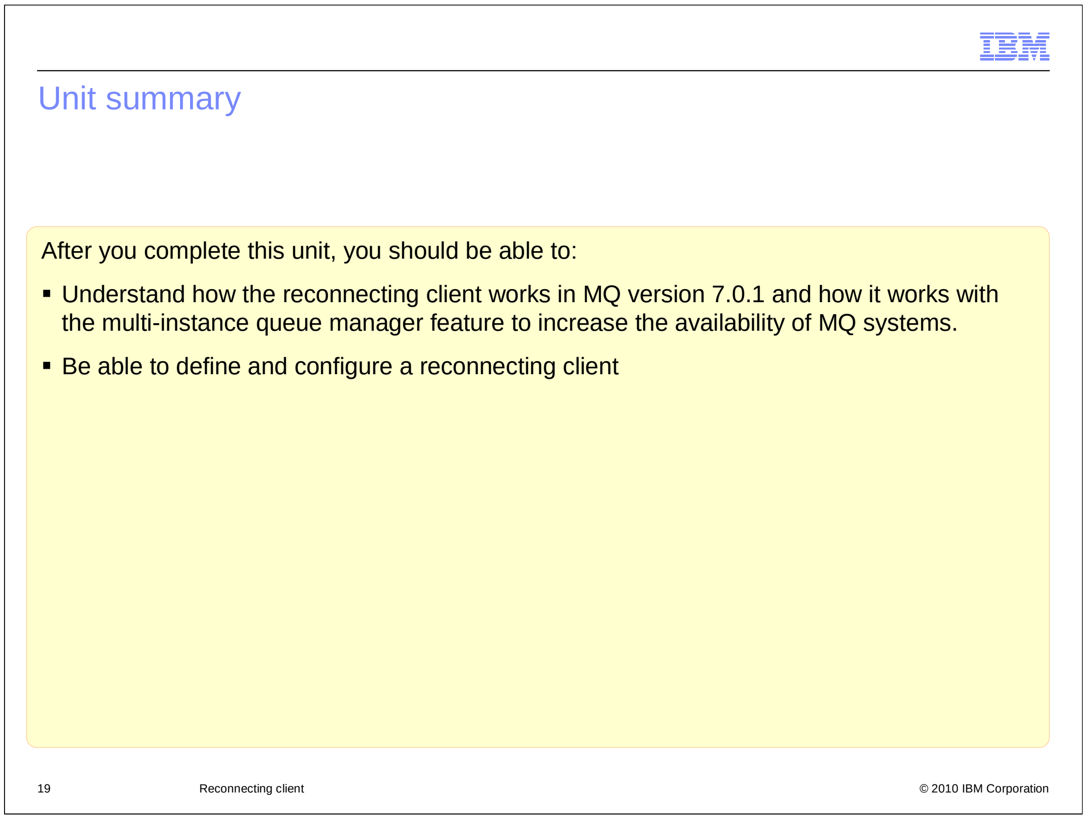


Reason code changes

- Should no longer see:
  - MQRC_CONNECTION_BROKEN
- New codes nested within a JMSException.
  - MQRC_RECONNECT_FAILED
    - MQ failed while attempting to reconnect your connection
  - MQRC_RECONNECT_QMID_MISMATCH
    - MQ reconnected your connection, but did not find the queue manager it was expecting

18

New reason codes may be returned nested inside the JMS exception. The first one shown indicates that a reconnect attempt failed for some reason. The second is the case where a reconnect to the same queue manager was specified, but the queue manager found at the host address was not that queue manager.



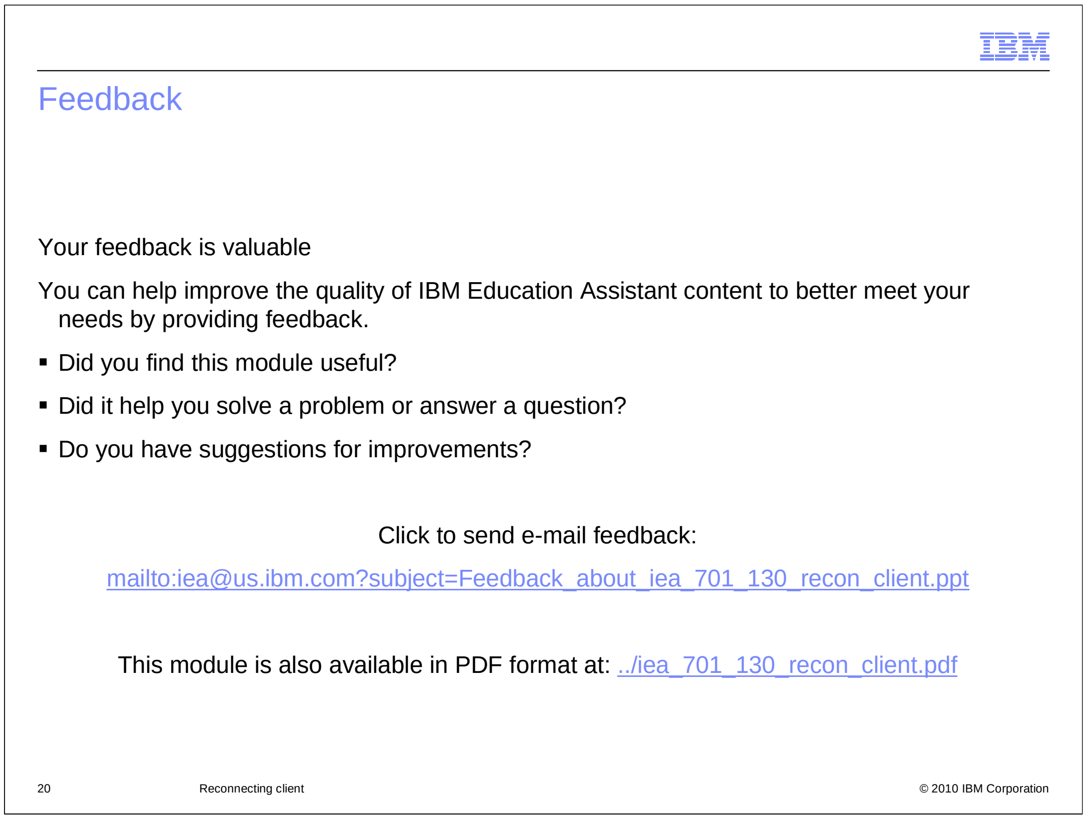
Unit summary

After you complete this unit, you should be able to:

- Understand how the reconnecting client works in MQ version 7.0.1 and how it works with the multi-instance queue manager feature to increase the availability of MQ systems.
- Be able to define and configure a reconnecting client

Now that you have completed this unit, you should have some understanding of how MQ 7.0.1 reconnecting client function works and how it is configured in both an MQ API environment and for JMS.

This unit did not attempt to cover the full range of syntax and options available, for which you should refer to the product information center.



Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

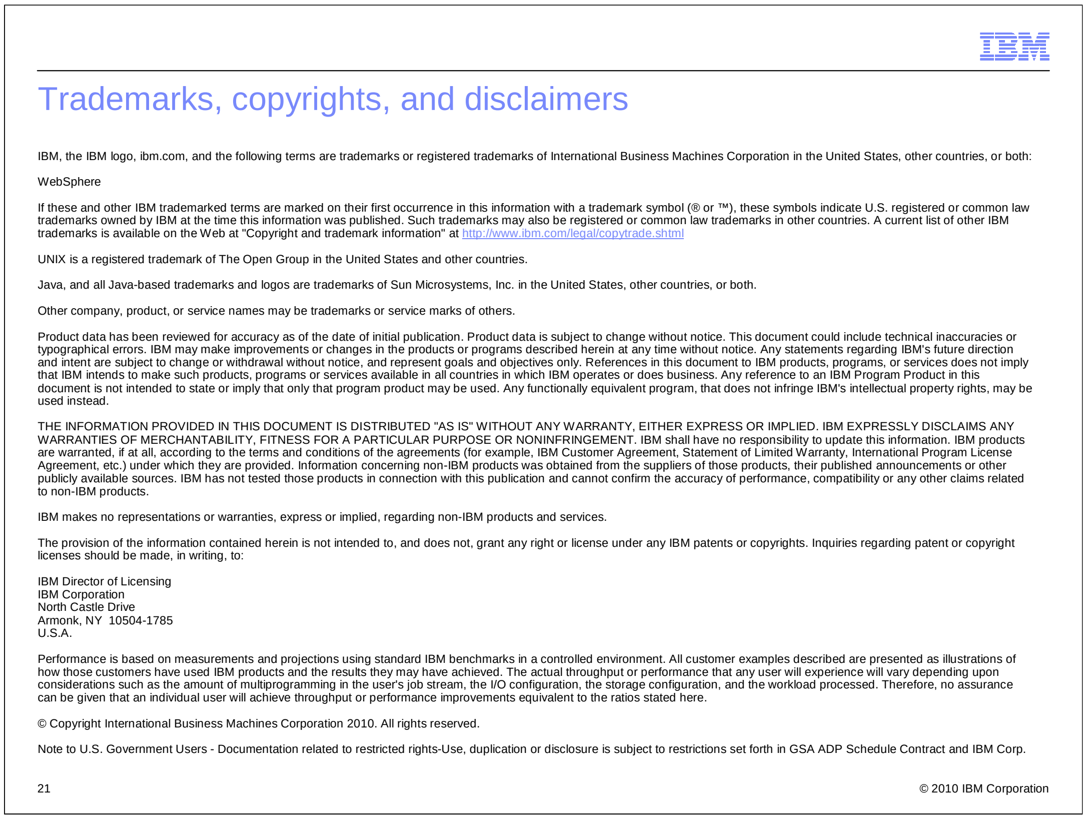
Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_iea_701_130_recon_client.ppt

This module is also available in PDF format at: ../iea_701_130_recon_client.pdf

20 Reconnecting client © 2010 IBM Corporation

You can help improve the quality of IBM Education Assistant content by providing feedback.



Trademarks, copyrights, and disclaimers