



IBM Software Group

## WebSphere® MQ V7.0

### *Client enhancements: conversation sharing and connection limiting*



@business on demand.

© 2008 IBM Corporation  
Updated October 14, 2008

This unit covers two of the client enhancements in WebSphere MQ version 7.0, conversation sharing and connection limiting.

## Unit objectives

After completing this unit, you should be able to:

- Describe the concept of conversation sharing.
- Know where conversation sharing can be used.
- Understand the benefits of reducing the number of sockets and threads.
- Describe the concept of channel instance limits.
- Know where channel instance limits can be used.



After you complete this unit, you should have some understanding of both conversation sharing and channel limits.

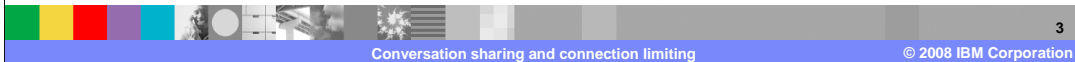
You will know how to control the conversation sharing and how to limit the number of clients connecting through SVRCONN channels.

You will also understand the potential benefits in using these features.

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.

## Conversation sharing

- In MQ V6, if a client process made multiple connections to the same queue manager
  - ▶ It used different sockets for each connection
  - ▶ The queue manager used a separate thread for each connection
  
- In MQ V7, introducing client multiplexing or conversation sharing.
  - ▶ Multiple conversations can flow down the same TCP/IP socket.
  - ▶ Only one queue manager thread will be used for the socket.
  - ▶ A heartbeat can be maintained in both directions at all times.

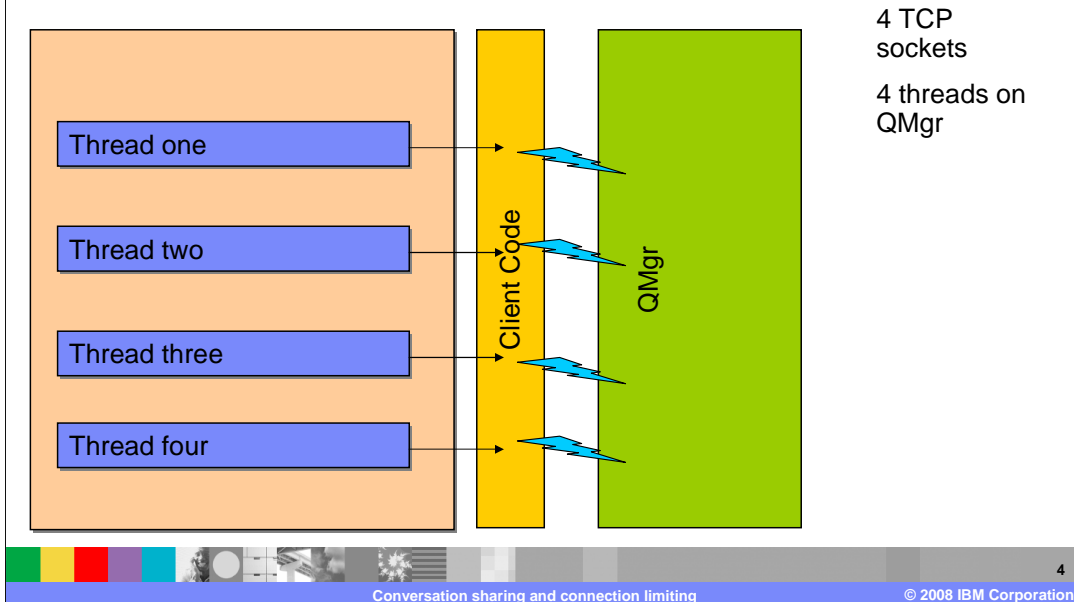


In older versions of MQ, if a client process made multiple connections to a queue manager, then each connection was essentially separate. A TCP socket pair was required for each connection and a thread on the queue manager was required for each connection. When each connection started or stopped the TCP session had to be created or destroyed; this can be expensive if SSL encryption was used.

In version 7.0 conversation sharing is introduced. When an application connects using the version 7 client code to a version 7 queue manager the client can hold a number of connections (or conversations) over the same TCP connection.

This will reduce the number of TCP resources used on both the client and queue manager system, fewer threads are required on the queue manager, and a continuous heartbeat is maintained between client and queue manager.

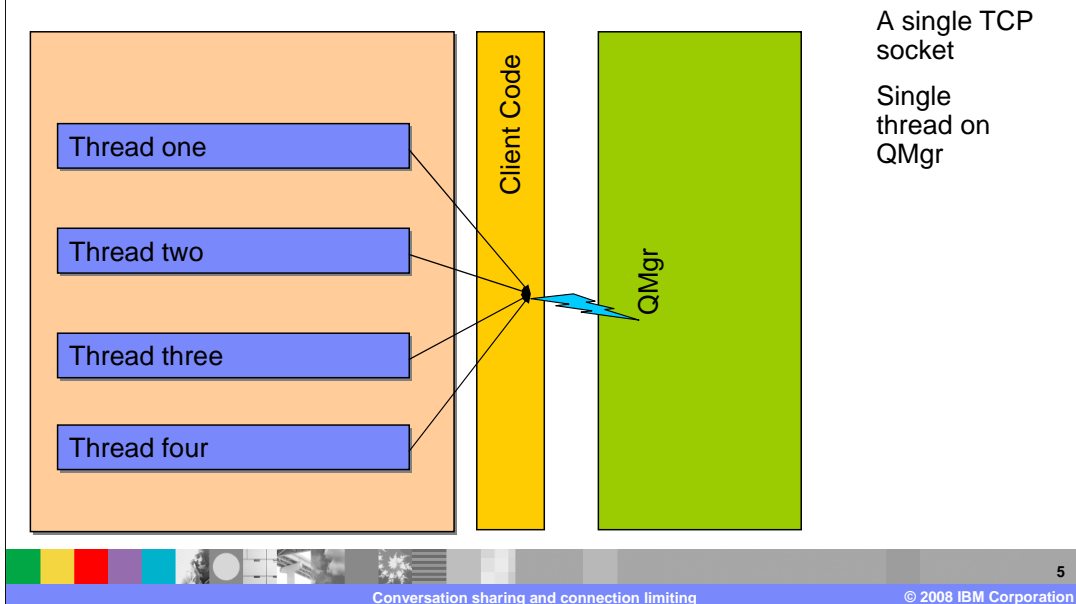
## MQ V6 multithreaded clients



In this illustration a multithreaded client application, for example the MQ JMS implementation, is communicating to a remote queue manager.

Four TCP socket pairs are required by the application. Each client thread requires a thread on the queue manager. Failure of the communications link between the systems would be detected by each socket pair independently and only when the client application tried to use the link.

## MQ V7 multithreaded, multiplexing clients



In version 7 the four threads in the client application share a single TCP socket to communicate to the queue manager.

A single thread on the queue manager is dealing with all the communication to the client. Loss of the communication link will be detected in a timelier manner by both the queue manager and the code client. The client code may only be able to reflect this to the client application thread when it next makes an MQ call.

## Shared conversations summary

- TCP/IP only; CLNTCONN / SVRCONN only
- On the QMgr end of channel, conversations are handled by one receive thread.
- Channel setup flows and security exchanges (including SSL) occur once per socket.
  - ▶ Lightweight start-up for subsequent conversations on the socket
- Only gives benefit for multi threaded applications.
  - ▶ The jms implementation is a primary beneficiary.
  - ▶ Jms sessions can now share a single TCP socket
- Heartbeats from both ends of a client channel at all times
  - ▶ Administrator STOP CHANNEL MODE(QUIESCE) is communicated immediately to the client (though not necessarily to the application program)
- It is the default in version 7.0



Here are the highlights for shared conversations.

This is a TCP only feature. No change has been made to LU6.2 or NETBIOS channels. It is a feature of client application connections so it applies only to the SVRCONN and CLNTCONN channel definitions.

Channel setup and security exchanges need only to occur on the first connection between a client and a queue manager. Subsequent connections sharing the same socket may incur much lower costs.

The benefits only really apply to multithreaded applications. This does however include the JMS implementation.

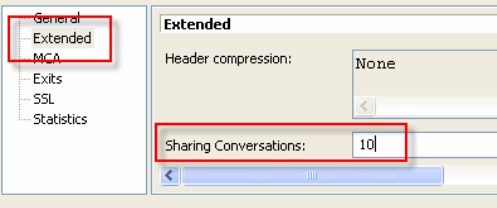
Heartbeating is occurring continuously across a channel at all times. This is one feature that non multithreaded applications can benefit from.

Shared conversations are the default in MQ version 7.0

IBM Software Group IBM

## Shared conversation definition

### Explorer



### RUNMQSC

```
define channel (sample)
  chlttype(SVRCONN)
  sharecnv(10)
```

- **Level of conversation sharing allowed:**
  - 0 : run as at V6
  - 1 : only one conversation on the socket, but it supports V7 features
  - Nnn : nnn is the maximum conversations per socket
  - Default is 10

7

Conversation sharing and connection limiting © 2008 IBM Corporation

The level of conversation sharing can be set for a channel using MQ explorer or the RUNMQSC application. Or indeed using PCF commands.

In explorer the properties are part of the “Extended” properties for a channel, in runmqsc the relevant keyword is “sharecnv”.

However set, the properties have the these meanings;

0 (zero) means this channel does not use the shared conversation feature at all; clients will behave just as in version 6.

1 (one) means that only a single connection per TCP socket is allowed, but the new heartbeating code will run for the channel.

Another integer means that number of conversations is allowed per channel instance.

So consider an application that starts ten threads (for example a JMS application with ten sessions) on a channel.

If sharecnv is set to zero or 1 then ten TCP socket sessions (and channel instances) are required.

If sharecnv is set to four, three TCP socket sessions (and channel instances) are required. The first four connections will share the first TCP socket session. Connections five through eight will share a second and the ninth and tenth will share a third.

It is important to realize that the shared conversation value is not a limit to the number of connections allowed, but only determines how many connections are supported on each TCP socket session and how many channel instances are required.

## Shared conversation status

### Explorer – Channel Status

Channel name	Channel type	Channel status
SAMPLE.SVRCONN	Server-connection	Running

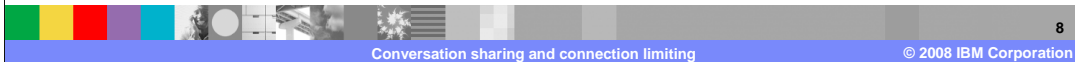
Issuer name	Max Conversations	Current Conversations
	10	1

### RUNMQSC DISPLAY CHSTATUS

```

dis chstatus(*) curshcnu maxshcnu
32 : dis chstatus(*) curshcnu maxshcnu
AMQ8417: Display Channel Status details.
CHANNEL(SAMPLE.SVRCONN)          CHLTYPE(SVRCONN)
CONNNAME(127.0.0.1)              CURRENT
STATUS(RUNNING)                  SUBSTATE(RECEIVE)
CURSHCNU(1)                       MAXSHCNU(10)
  
```

- For each Channel instance
  - ▶ Maximum number of shared conversations (the default is 10)
  - ▶ Current actual number on shared conversations on this instance



The channel status displays from both explorer and RUNMQSC displays the actual number of conversations being run on each channel instances at this time.

The channel status display in explorer lists the “Max conversations” and the “Current Conversations” for each channel instance running. In a similar manner the “display chstatus” command (or equivalent PCF command) uses the keywords “MAXSHCNU” and “CURSHCNU” to return the same data.



## Shared conversation coding

- For most applications use of conversation sharing is transparent to the application.
- Multi threaded application can use MQCONNX MQCNO
  - ▶ MQCNO\_NO\_CONV\_SHARING
    - Only one conversation per socket,
    - Still supports V7 heartbeating.
  - ▶ MQCNO\_ALL\_CONVS\_SHARE (default)
    - All types of connection can share the socket
- Exits – **Channel exits may need reviewing !**
  - ▶ A new boolean field MQCXP SharingConversations is passed to exits.
  - ▶ This to FALSE on first or only conversation and TRUE on subsequent conversations using channel.



Most applications using either the MQI or JMS will not need to be aware of the conversation sharing features. But for those MQI applications that use the extended connect verb MQCONNX they can force the application to use only one conversation per channel instance by specifying the MQCNO\_NO\_CONV\_SHARING option.

One area where code changes may be needed is channel exits. Additional parameters are available to channel exits in versions 7.

In particular, when an exit is called and conversation sharing is happening, the second and subsequent calls to initialize the exit will be sharing some data with the first call. If exits update this data then they may need modifying. Exits that perform security checks or journaling may take different actions for the first and subsequent conversations on a channel instance.

## Shared conversation with JMS

- The MQ JMS provider is a major multithreaded application that can benefit from conversation sharing.
- The connection factories provided by MQ have a new parameter SHARECONVALLOWED.
- If this is set to YES and the channel supports shared conversations then the MQ for JMS classes will be using shared conversations.

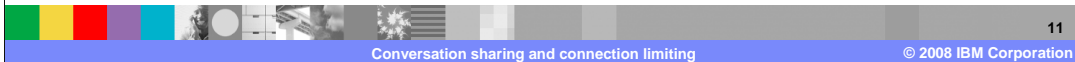


The MQ supplied connection factories for JMS objects now have a property SHARECONVALLOWED that specifies whether connection sharing is to be used for these connections. Connection sharing will occur when this is YES and the channel supports connection sharing.

## Benefits of shared conversations

- With very many shared conversations, thread usage on the queue manager is dramatically reduced
- Failures of connectivity are discovered quickly (better heartbeating)
- Quiescent stop works more effectively

After migration, conversation sharing is enabled



The benefits of connection sharing are reduced resource consumption in the form of threads and TCP sessions. This will be reflected in a reduced number of active channel instances reported in explorer displays and other monitors.

Communication failures will be reported in a timelier manner and the Quiesce of channels will work more effectively.

It is worth noting that after migration from previous versions conversation sharing is enabled on the migrated system.

## Channel instance limits

- In version 7.0 limits can be defined on the number of instances of a server connection channel that can be started on a queue manager.
- These limits will help stop a queue manager being disrupted by a large number of connects from rogue or failing client applications.
- The maximum total number of instances of a SVRCONN channel can be specified.
- The maximum number of instances from a single client (IP address) can be specified.



The other topic covered in this presentation is about setting limits on the number of client channel instances that can be started.

A queue manager system may be designed to run with some particular maximum number of client connections, say 100, and the system hardware and software may have been designed around that number. If suddenly many more connections are made say 10,000, the system may be overwhelmed and fail due to lack of resources. These additional connection attempts may be caused by a software bug, or by malicious intent – a denial of service attack.

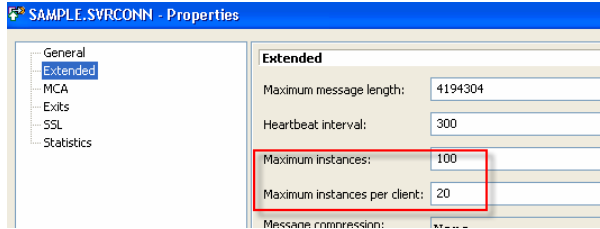
In MQ version 7.0 the SVRCONN channel can have limits set so that connections are refused in a predictable manner by the queue manager.

For each SVRCONN channel the total number of channel instances permitted can be specified and the maximum number of channel instances that can be started from a single client – IP address.

For example, limit the total number of client connections to 150 (50 percent above expectation) and no more than two connections from any one IP address.

## Channel limits

### Explorer



### RUNMQSC

```
define channel (sample)
  chltpe (SVRCONN)
  maxinst(100)
  maxinstc(10)
```

The channel limits can be specified using MQ Explorer or RUNMQSC commands or the equivalent PCF commands.

In explorer the settings are found in the “extended” tab of the channel definition.

The DEFINE CHANNEL command has the equivalent parameter MAXINST and MAXINSTC.

## Unit summary

Having completed this unit, you should be able to:

- Describe the concept of conversation sharing
- Know where conversation sharing can be used
- Understand the benefits of reducing the number of sockets and threads
- Describe the concept of channel instance limits
- Know where channel instance limits can be used



This completes the introduction to the client conversation sharing and channel instance limits.

It has covered how conversation sharing can reduce the TCP resources required by multithreaded client applications and how setting channel instance limits can protect the queue manager from being overwhelmed by a failing or malicious application.

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\\_710\\_wmqv7\\_Client\\_Multiplexing.ppt](mailto:iea@us.ibm.com?subject=Feedback_about_iea_710_wmqv7_Client_Multiplexing.ppt)

This module is also available in PDF format at: [../iea\\_710\\_wmqv7\\_Client\\_Multiplexing.pdf](..iea_710_wmqv7_Client_Multiplexing.pdf)



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

## Trademarks, copyrights, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM                      WebSphere

A current list of other IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

© Copyright International Business Machines Corporation 2008. All rights reserved.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.