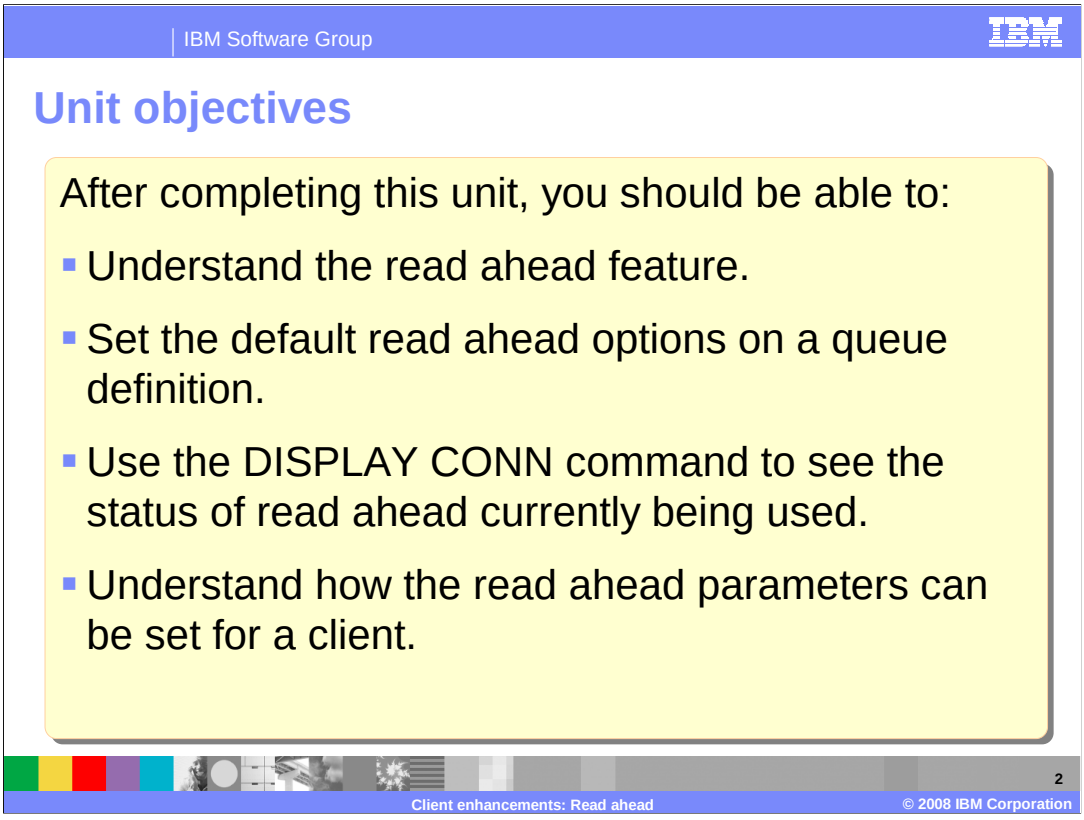
This unit covers the major enhancement available to TCP client connected applications getting messages from MQ. In some circumstances significant throughput improvement can be gained



After you complete this unit, you should understand the “read ahead” feature of MQ version 7.

Understand how the default read ahead values can be specified for a queue.

Code the options of the MQOPEN and MQSUB calls that enable read ahead.

Know that the connection status shows whether read ahead is taking place for a given connection.

Know how the client configuration file can be used to change the defaults for read ahead.

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.

Introduction

- One of the strengths of WebSphere MQ is, and remains, its ability to assure once and once only message delivery.
- Such high qualities of service are not required for all applications.
- Read ahead is a feature that allows messages being read from a queue to be sent by a queue manager to a client in advance of the MQGET being issued.
- The feature is relevant in a client rather than bindings connection where high quality of service is not needed.
- In the right circumstances significant performance improvement should be seen.
- It is enabled by a combination of API settings and queue / topic definitions.
- The feature is available to JMS clients also.

One of the strengths of WebSphere MQ is, and remains, its ability to assure once and once only message delivery.

Such high qualities of service are not required for all messages in all applications.

Using the read ahead feature allows an application that is making a sequence of MQGET requests to gain a faster throughput. This is accomplished by the MQ server code and the MQ client code cooperatively streaming likely messages to the client side before the application actually issues a particular MQGET. This can result in a very fast delivery of messages to the application's eventual MQGET request.

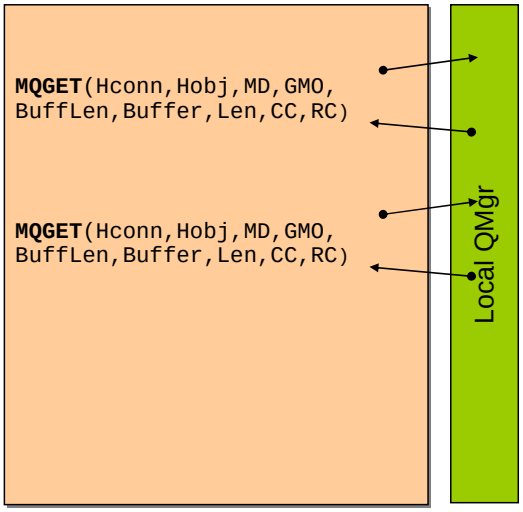
The feature is relevant in a client rather than bindings connection where a high quality of service is not required.

In the right circumstances significant performance improvement should be seen.

It is enabled by a combination of API settings and queue definitions.

The benefits are available to all clients, including JMS applications.

MQGET in binding mode

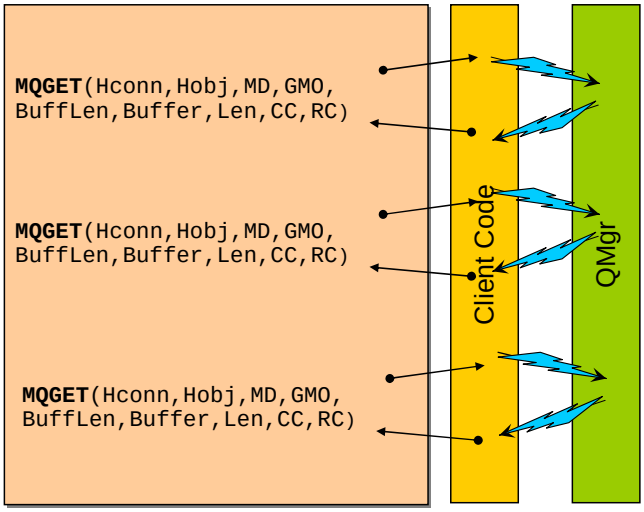


- Each message is requested from the queue manager and passed to the application.

Consider a typical MQ application that is performing a sequence of MQGET operations. If this application is connected to a queue manager on a local machine in “binding” mode then this slide illustrates what happens.

For each MQGET operation: The application program begins the MQGET Call; the request is passed to the queue manager which retrieves the message and passes the content back to the caller in a buffer; at which point zero completion and reason codes are passed back to the application.

MQGET in V6 client mode



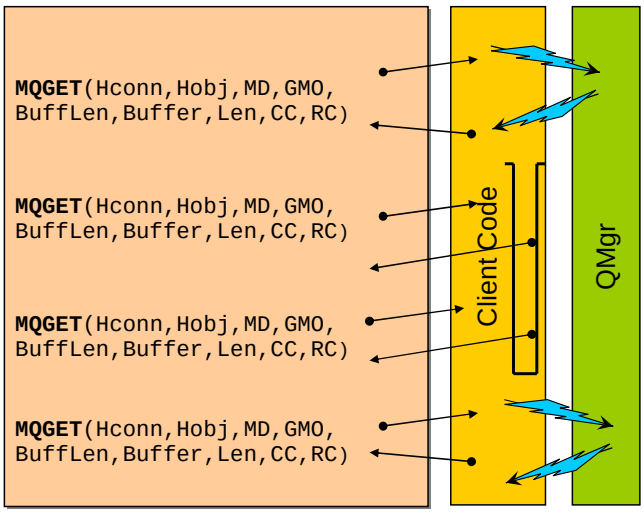
- Each message request flows over the network to the queue manager. The message is then returned over network and passed to the application.

In a version 6 (or earlier) client application connected over a TCP communications link very much the same thing happens with additional steps.

For each MQGET operations: The application program begins the MQGET Call; the request is passed to the MQ client code; the MQ client code passes the request over the communication link using TCP/IP protocols to the queue manager. The queue manager retrieves the message and delivers the message content to the MQ client code over the TCP/IP link. The MQ client code then passes THE MESSAGE AND RESPONSE CODES to the application.

In this case the application code thread is held up blocked while the request is flowed to the queue manager, the message is retrieved and the message and response flowed back.

MQGET in client mode with read ahead



- Client code reads multiple messages from the queue manager.
- Those messages are stored in memory for delivery to the application.

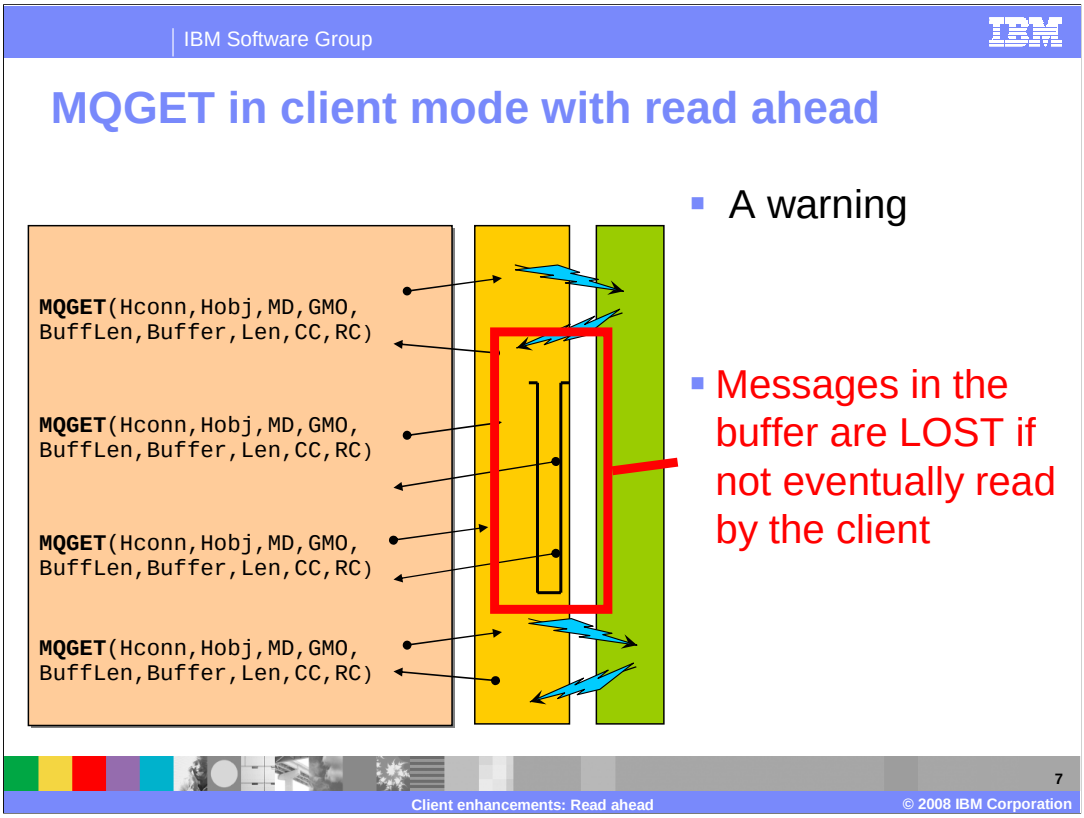
In a version 7 client application connected over a TCP communications link to a version 7 queue manager something rather different can occur when the read ahead feature is enabled.

The application program begins the MQGET call and the request is passed to the MQ client code. Now the MQ client code makes a request to the MQ server passing the MQGET parameters. When the first qualifying message is returned to the MQ client code, it is passed to the application, together with completion code and all other properties. However now the MQ client code and the MQ server continue communicating and if other messages are available which would satisfy the initial request they (or some of them) are passed up to the MQ client. The MQ client stores these messages in a memory buffer ready for rapid delivery to the requesting application.

If the application issues another suitable MQGET the message can be satisfied from the MQ client's memory buffer giving a very rapid response. In this way the application is able to issue a sequence of MQGET requests without having to wait for networking operations to be carried out for each request.

As the application code issues more suitable MQGETs messages are retrieved from the MQ client buffer, in parallel the MQ client code and the MQ server continue to communicate and add additional messages to the buffer. One read ahead buffer is maintained for each open queue handle.

This feature works best when the application design is one where the program makes a sequence of similar MQGET requests and is designed to process all the suitable messages on a QUEUE.



At this point it is important to be clear about one thing. When messages are retrieved by the queue manager and MQ client code and placed in the read ahead buffer, they are permanently removed from the queues where they were stored.

Messages can be lost (removed permanently from the queue manager and not passed to an application program for processing) if the client application terminates with messages still in the MQ client buffer. This can happen through a hardware or software failure or by the program closing the queue with messages still in the buffer. Later slides show new MQCLOSE options that can detect this condition.

The read ahead option is only suitable for non persistent messages where message loss is not critical.

Most installations do have such messages. For example the MQ Explorer communicates with queue managers using queues to return messages listing the queues on a queue manager. The response messages on the queue SYSTEM.COMMAND.RESPONSE.QUEUE. If explorer does not get the responses in a timely manner it does not need them at all.

New MQOPEN options

- New options in MQOO
- **MQOO_NO_READ_AHEAD**
  - Assures the current (V6) behavior
- **MQOO_READ_AHEAD**
  - Non-persistent messages may be sent to the client ahead of an application requesting them.
  - Read ahead does not always happen. For example: the queue may be disabled for read ahead.
- **MQOO_READ_AHEAD_AS_Q_DEF**
  - The option to be used is that implied by the default read ahead setting on the queue definition.
  - Default value
- The options are only valid for local, alias, and model queues.

Whether or not messages are being retrieved with read ahead in operation, there is a combination of new options on the MQOPEN and new attributes of a queue definition.

The new options are specified in the open options (MQOO) block of an MQOPEN call.

Possible values are MQOO_NO_READ_AHEAD, this setting assures the version six behavior. It would be specified in an application that had to be assured that no intermediate buffering of messages by the MQ client code was to occur. You would specify this if that behavior was essential to the logic of the application.

MQOO_READ_AHEAD would be specified if this application was suitable for read ahead. Read ahead will occur unless explicitly DISABLED by the queue definition.

The default value, and so the one used by all existing applications, is MQOO_READ_AHEAD_AS_Q_DEF. This means that the decision is left the administrators of the system and the behavior will be that specified by the DEFREADA attribute.

These options can only be applied to local, alias and model queues, and only when an MQ version 7 client is connected to an MQ version 7 queue manager.

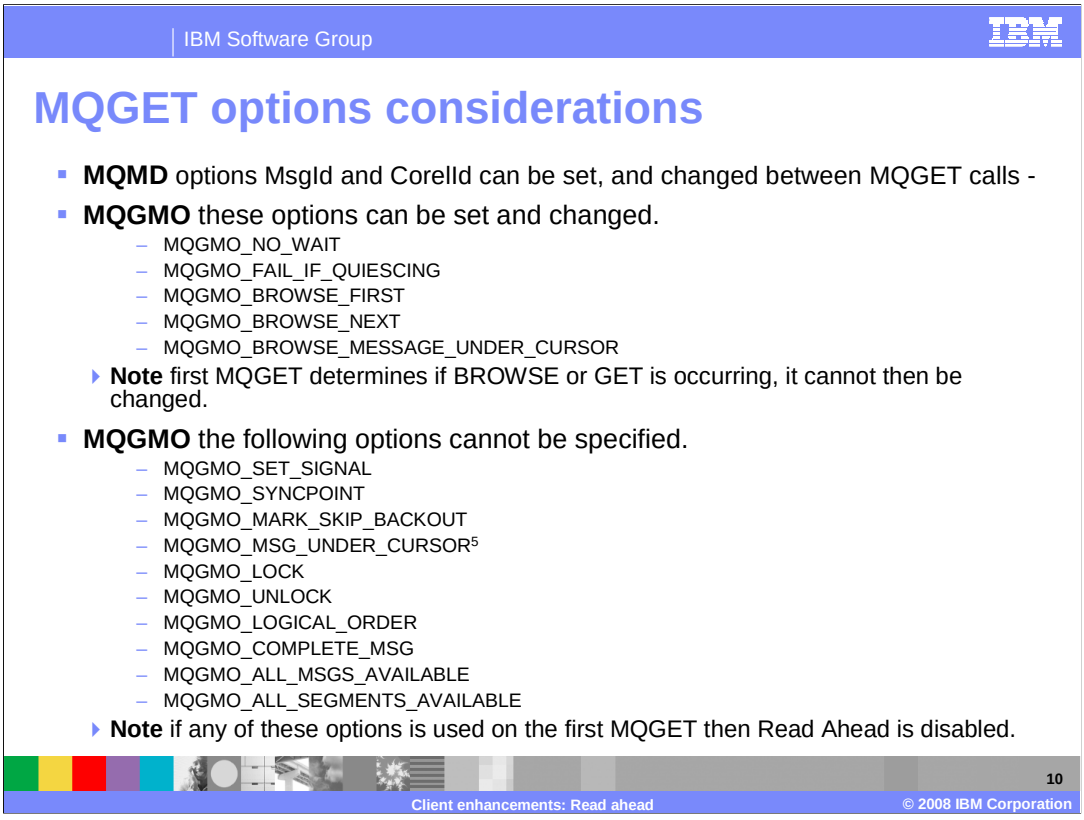
New MQSUB considerations

- The options described for MQOPEN also apply to MQSUB where a managed handle is being created.
  - Though they are “spelled” MQSO_... In this case.
- The properties specified then apply to the managed queue created to receive the messages for the subscription.

Remembering that one thing an MQSUB call can do is to create and open a managed queue to receive the messages resulting from a subscription. It is not surprising to find that the MQOPEN options relating to read ahead also apply to the MQSUB Call.

The options are prefixed with MQSO rather than MQOO but have the same meaning.

MQSO_READ_AHEAD_AS_QDEF would in this case mean as the model queue definition that was used as a basis for the queue definition.



As mentioned earlier the read ahead feature works best when an application is reading sequentially all the suitable messages on a queue. But the behavior of MQ when the read ahead feature is used in other circumstances has to be defined.

In the message descriptor the message ID and the correlation ID can be changed between calls and the read ahead behavior continues. But note: Suppose you perform an MQGET Corelid=A and three records are placed in the read ahead buffer, one of which is returned on the call. The next MQGET is for a corelid=B, then additional messages are fetched from the queue manager and returned. But the two corelid=A messages have been removed permanently from the queue manager and will be lost unless this application goes back and reads them.

In the Get Message options the indicated options can be specified and changed between calls. Note that the decision between performing browse or destructive get is made on the first call and cannot be changed.

The remaining get message options listed are not permitted in conjunction with read ahead.

New MQCLOSE options

- It is possible that when closing a queue messages are still in the Read Ahead buffer.
- New options in MQCO
- **MQCO_IMMEDIATE**
  - Any messages in the read ahead buffer are **discarded**.
  - Default value
- **MQCO_QUIESCE**
  - If messages exist in the read ahead buffer MQCLOSE call will return with a **warning** of **MQRC_READ_AHEAD_MSGS** and the queue handle will remain valid.
  - No more messages will be added to the read ahead buffer, though the remaining messages may be read.
  - Programmer response is to consume the messages until an MQCLOSE with MQCO_QUIESCE option is successful.
- The options are only valid for queues.

Because it is possible that an MQCLOSE operation might occur when unread messages are still held in the MQ client read ahead, new options are added to the MQCLOSE call.

The default value is MQCO_IMMEDIATE. This is a close that closes the queue and discards any messages in the read ahead buffer with no warning. Be warned.

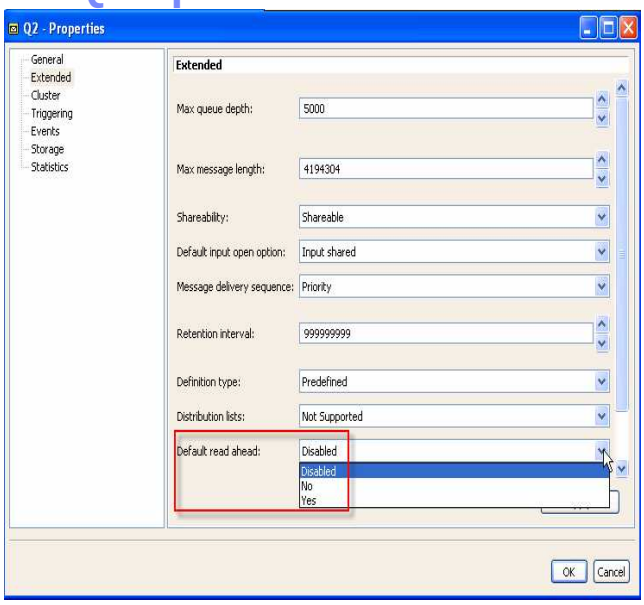
This is made the default because it adds no new return codes to the MQCLOSE so that existing applications will not have to code for new return codes.

The alternate value of MQCO_QUIESCE will cause the close to only succeed if no messages remain in the read ahead queue.

If messages remain an MQCC code of “warning” will be set, together with a reason code of MQRC_READ_AHEAD_MSGS and the queue handle remains open. After this call the MQ server and MQ client will add no more messages to the read ahead queue but will allow the remaining messages to be retrieved using MQGET. The application can now get the remaining messages, and MQRC_NO_MSG_AVAILABLE will be returned when no more messages remain in the buffer. Another MQCLOSE with the MQCO_QUIESCE option should then be made.

Queue definition in MQ explorer

- In the extended tab



As mentioned in an earlier slide the read ahead feature is enabled by a combination of MQOPen (or MQSUB) options and queue definitions.

In MQ explorer the default read ahead option is found on the extended tab of queue definition.

The allowed values are “yes”, “no” and “disabled”. “Yes” and “no” values are the defaults when the MQOO_READ_AHEAD_AS_Q_DEF option is used but the “disabled” option means that read ahead will not occur even if the application program uses the explicit read ahead option.

Queue definition default read ahead

- Keyword DEFREADA values NO, YES or DISABLED.
- For QLOCAL, QMODEL, QALIAS

```
    30 : display q(q2) defreada
AMQ8409: Display Queue details.
   QUEUE(Q2)                                TYPE(QLOCAL)
   DEFREADA(NO)
```

```
alter qlocal(q2) defreada(disabled)
    31 : alter qlocal(q2) defreada(disabled)
AMQ8008: WebSphere MQ queue changed.
display q(q2) defreada
    32 : display q(q2) defreada
AMQ8409: Display Queue details.
   QUEUE(Q2)                                TYPE(QLOCAL)
   DEFREADA(DISABLED)
```


Using RUNMQSC the option is spelled “DEFREADA”, and takes values “NO”, “YES” and “DISABLED” with the same meanings as in MQ Explorer.

The option is valid for local, model and alias queues.

When an application is reading messages from a managed queue created as a result of subscribing to some topic the default read ahead for that queue is determined by the relevant model queue definition.

DISPLAY CONN change

- Keyword READA can be specified on DISPLAY CONN Command to show whether Read Ahead is in force.
- Values can be **NO, YES, INHIBITED, BACKLOG**

```
display conn(*) TYPE(HANDLE) READA
     37 : display conn(*) TYPE(HANDLE) READA
AMQ8276: Display Connection details.
```

```
MQ8276: Display Connection details.
   CONN(2CD9034720002501)
   EXTCONN(414D5143574D513720202020202020202020)
   TYPE(CONN)

   OBJNAME(WMQ7)                                 OBJTYPE(QMGR)
   READA(NO)

   OBJNAME(SYSTEM.ADMIN.COMMAND.QUEUE)           OBJTYPE(QUEUE)
   READA(NO)

   OBJNAME(AMQ.MQEXPLORER.898905492)             OBJTYPE(QUEUE)
   READA(NO)
```


Additional information is now available on the display of connection status.

When displaying a connection handle the read ahead status can be displayed. The keyword for RUNMQSC is READA.

The possible values are “YES”, “NO”, “INHIBITED” and “BACKLOG.”

“YES” means that the read ahead feature is being used on the connection.

“NO” means that the feature was not requested and is not being used.

“INHIBITED” means that the application requested read ahead but the queue definition has explicitly disabled read ahead.

Readahead “BACKLOG” status

- **BACKLOG**
- Read ahead of non persistent messages is enabled for the queue that the connection has open but is not being used efficiently.
- Possibly the client application consuming the messages has altered its selection criteria.
- Messages that were read ahead by the client with the previous selection criteria have not been consumed by the client application.
- These messages will remain in the read ahead buffer on the client. If a significant number of these messages are present they could prevent read ahead from operating efficiently and **BACKLOG** will be reported.

Read ahead of non persistent messages is enabled for the queue that the connection has opened but is not being used efficiently.

The client application consuming the messages has possibly altered its selection criteria. Messages that were read ahead by the client with the previous selection criteria have not been consumed by the client application. These messages will remain on the read ahead queue. If a significant number of these messages are present they could prevent read ahead from operating efficiently and BACKLOG will be reported.

If a client application design is not suited to read ahead, the MQOPEN option MQOO_NO_READ_AHEAD should be specified in the client application or the default read ahead value of the queue being opened altered to either NO or DISABLED.

Alternatively if the messages that were sent to the client with the previous selection criteria are no longer required, a configurable purge interval on the client can be set to automatically purge these messages from the client. This is specified in the MQ client configuration file.

Controlling read ahead

- A new stanza in the client configuration file
  - MessageBuffer

```
MessageBuffer:
   MaximumSize=-1
   Updatepercentage=-1
   PurgeTime=0
```

- Three parameters.
- MaximumSize
  - Default – Client determines
- UpdatePercentage
  - Default – Client determines
- PurgeTime
  - Default – No purging

When read ahead is enabled the client will receive messages ahead of an application requesting them. These messages are stored in a read ahead buffer on the client. By default the client will determine a suitable size for this buffer, and will update the contents as appropriate. The size and updating of this buffer can be influenced by the setting of appropriate values in the client configuration file. The client configuration file used can be identified in a number of ways and can have any name, see the information center for details.

The client configuration file used by a client connection has a new “stanza” (a stanza is a group of parameters) controlling read ahead.

The “MaximumSize” parameter is the maximum size of the read ahead buffer in KiloBytes. The special value of minus one (-1) means the MQ client code determines the value, this is the default and recommended value.

The “UpdatePercentage” parameter is used to determine when the MQ client code should request more data from the server. The special value of minus one (-1) means the MQ client code determines the value, this is the default and recommended value.

The “PurgeTime” parameter determines when unread messages will be purged from the read ahead buffer. The special value of zero (0) means no purging occurs. (-1) means the MQ client code determines the value, this is the default and recommended value. If pugetime is specified as non zero, then it is the number of seconds after which messages may be purged by the MQ client code. Messages which have been moved to the read ahead buffer and not read by the client after “PurgeTime” seconds may be deleted from the read ahead buffer by the MQ client code. Again it should be stressed that such messages are lost. PurgeTime should only be set if your application design may leave messages in the read ahead buffer that are not required.

Unit summary

Having completed this unit, you should be able to:

- Understand the read ahead feature.
- Set the default read ahead options on a queue definition.
- Use the DISPLAY CONN command to see the status of read ahead currently being used.
- Understand how the read ahead parameters can be set for a client.

Having completed this unit, you should understand the “Read Ahead” feature of MQ version 7; understand how the default read ahead values can be specified for a queue, know how to code the options of the MQOPEN and MQSUB calls that enable read ahead, know that the connection status shows whether read ahead is taking place for a given connection and know how the client configuration file can be used to change the defaults for read ahead.

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.

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_730_wmqv7_Client_ReadAhead.ppt

This module is also available in PDF format at: ../iea_730_wmqv7_Client_ReadAhead.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.

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.