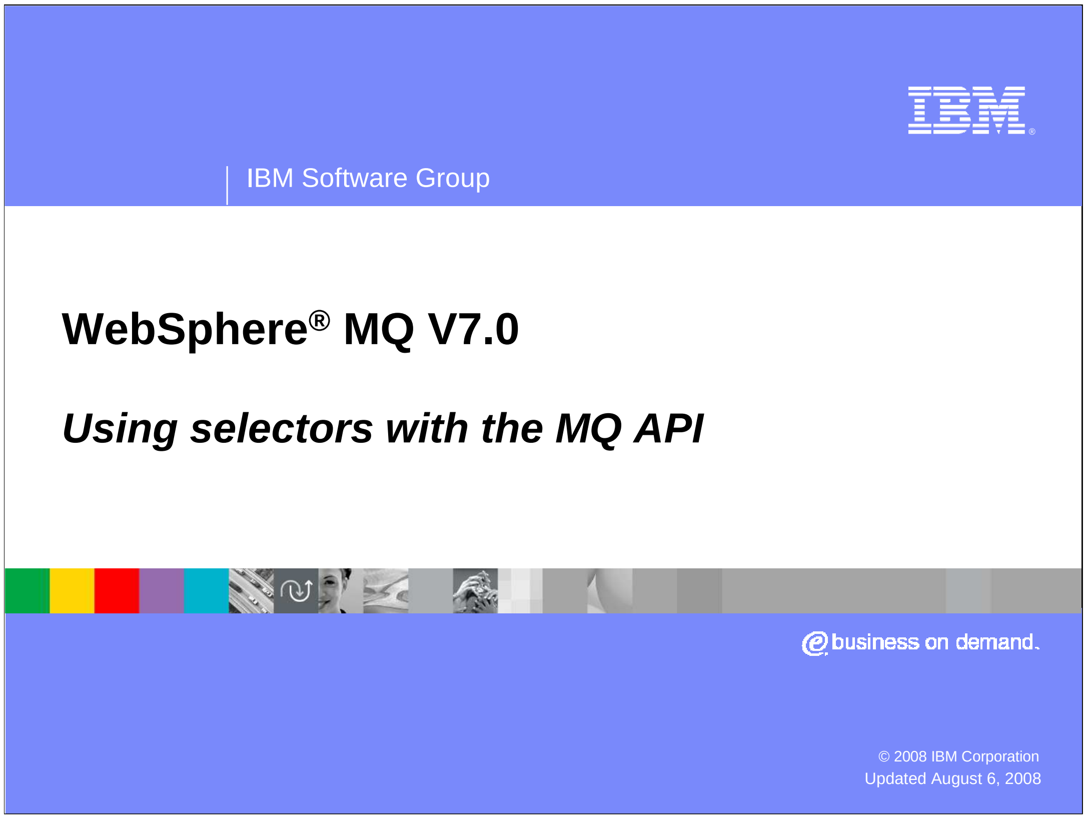


IBM Software Group

WebSphere® MQ V7.0

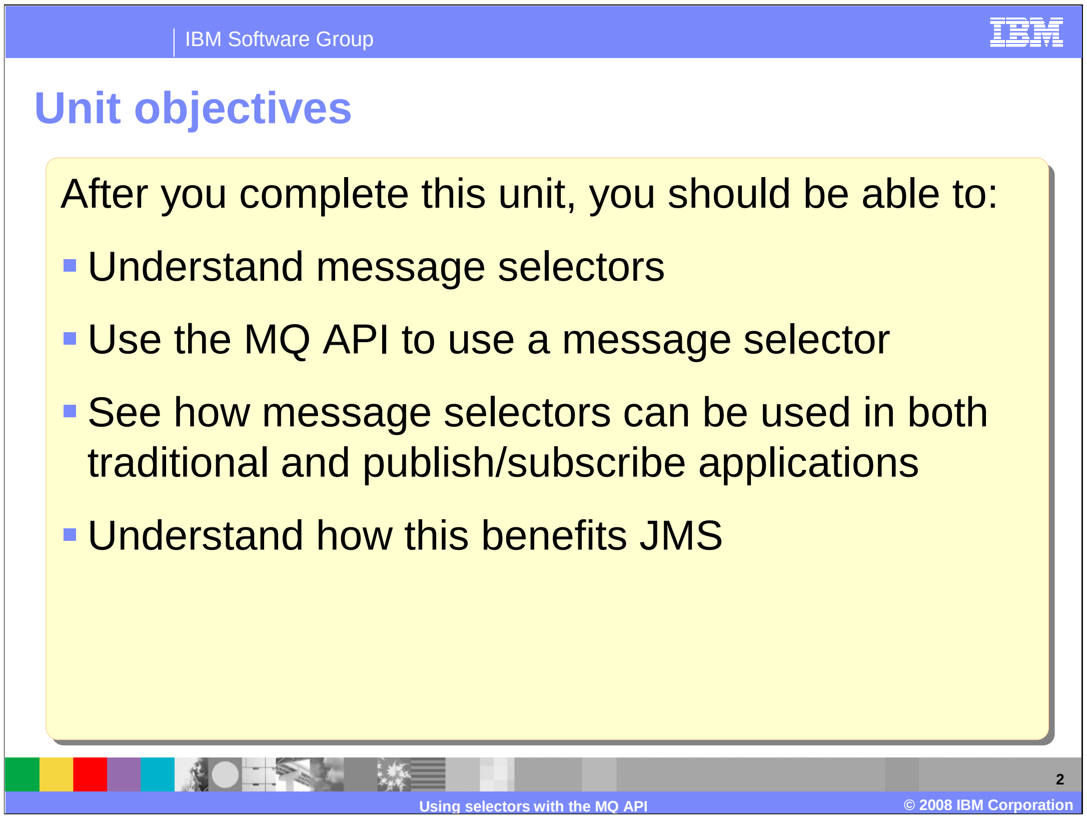
Using selectors with the MQ API

business on demand.

Updated August 6, 2008

This unit covers the MQ API (or MQI) enhancement in WebSphere MQ version 7. The new message selector feature added in this release is covered. Message selectors requires an understanding of Message Properties which is the subject of a separate presentation.

This unit assumes a reasonable understanding of the existing WebSphere MQ API for putting and getting messages to and from queues.



Unit objectives

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

- Understand message selectors
- Use the MQ API to use a message selector
- See how message selectors can be used in both traditional and publish/subscribe applications
- Understand how this benefits JMS

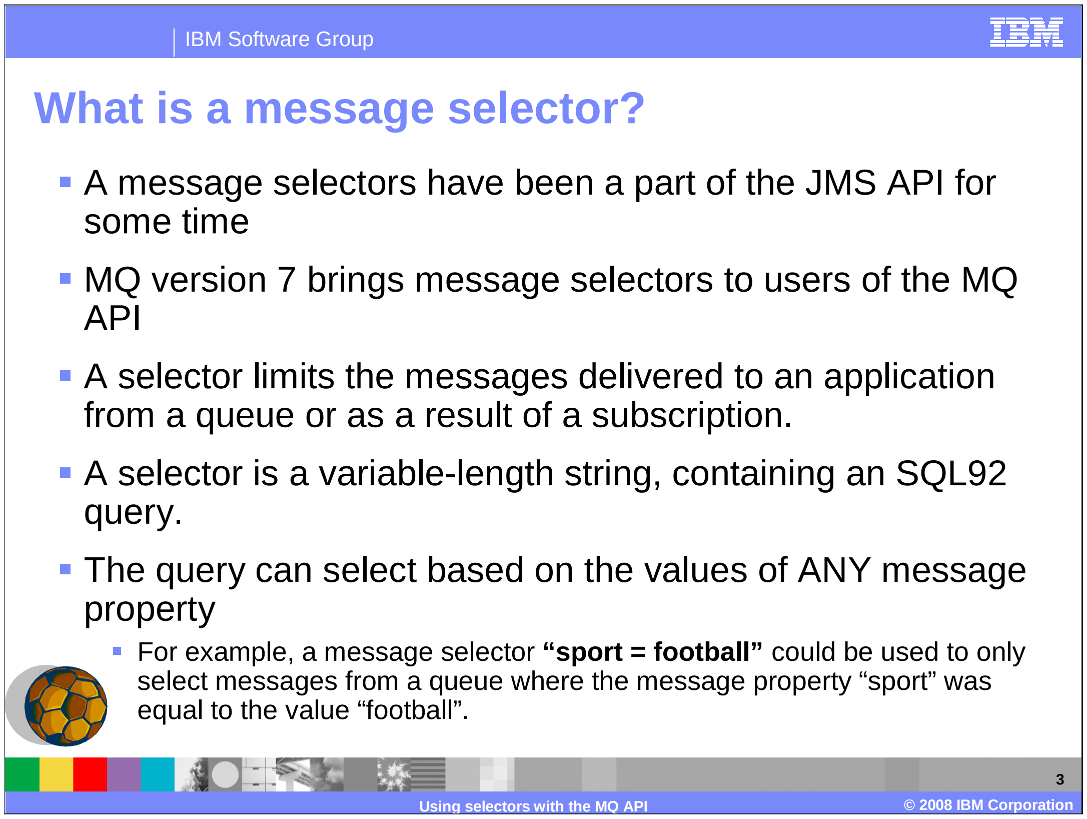
After completing this unit you should have some understanding of what message selectors are and why they have been introduced.

An overview is given of how selectors work and how they are coded as an extension to the MQOPEN call and in the new MQSUB call.

With the assistance of the information center you should be able to code MQ7 applications that use selectors to limit the range of messages delivered as a result of a subscription or got from a queue.

Although the JMS specification supported by MQ7.0 as JMS provider has not changed, these features enable a simpler, cleaner and more efficient implementation of JMS selectors by MQ.

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.



What is a message selector?

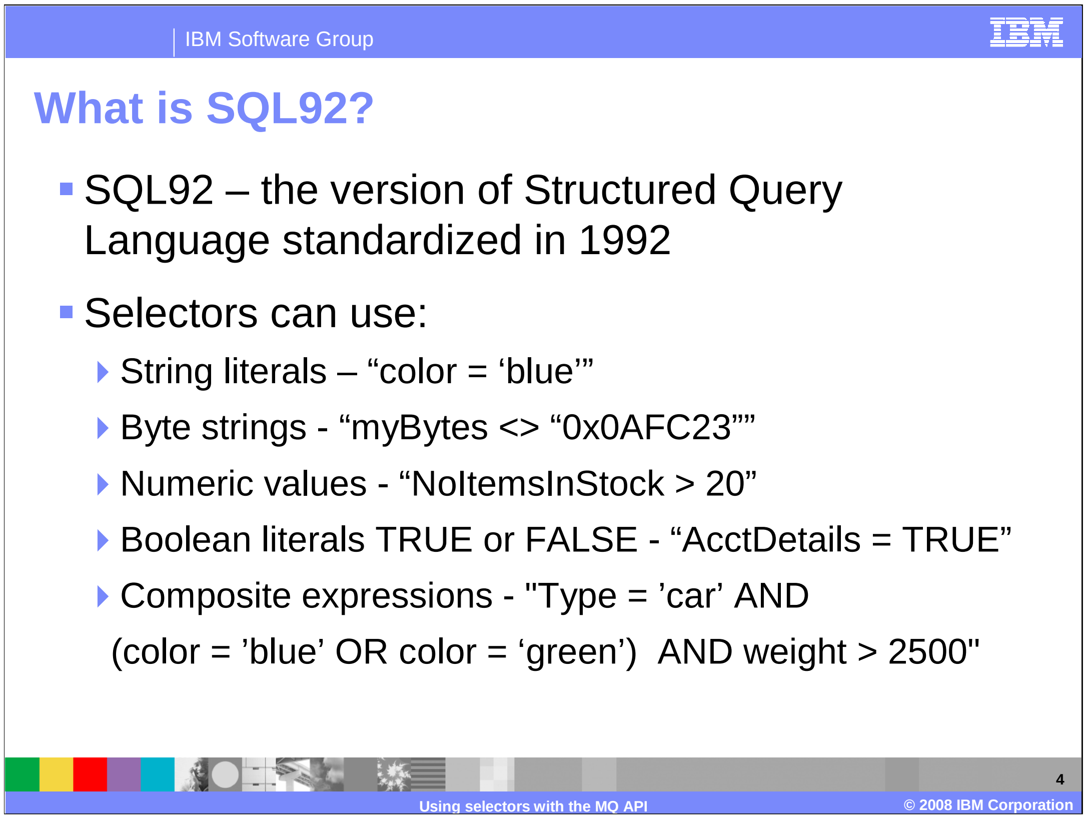
- A message selectors have been a part of the JMS API for some time
- MQ version 7 brings message selectors to users of the MQ API
- A selector limits the messages delivered to an application from a queue or as a result of a subscription.
- A selector is a variable-length string, containing an SQL92 query.
- The query can select based on the values of ANY message property
  - For example, a message selector **"sport = football"** could be used to only select messages from a queue where the message property "sport" was equal to the value "football".

A message selector is a concept that has been in the JMS specification for a long time. It is a way of limiting the messages that are passed to an application to those that meet certain criteria. Those criteria are based on the values of the message properties and only the value of the message properties. It is important to understand that selection cannot be based on any values of the message payload, only on the message property values.

The form of the selector is a string that contains a selection expression. The selection expression is based on the well known SQL92 specification for database queries. The expression will contain references to message properties and constants.

The example shown on the slide shows a selection string that limits messages to those that have a message property name "sport" **AND** where the value of that property is football.

The next slide illustrates some possible expressions that can be used for selection.


IBM Software Group

What is SQL92?

- SQL92 – the version of Structured Query Language standardized in 1992
- Selectors can use:
  - String literals – “color = ‘blue’”
  - Byte strings - “myBytes <> “0x0AFC23””
  - Numeric values - “NoItemsInStock > 20”
  - Boolean literals TRUE or FALSE - “AcctDetails = TRUE”
  - Composite expressions - "Type = ’car’ AND (color = ’blue’ OR color = ‘green’) AND weight > 2500"

4

Using selectors with the MQ API

© 2008 IBM Corporation

SQL92 is a standardized specification of the Structured Query Language used to query relational databases. It is used for selectors in the JMS specification and the same definition is used in the MQ API implementation of selectors.

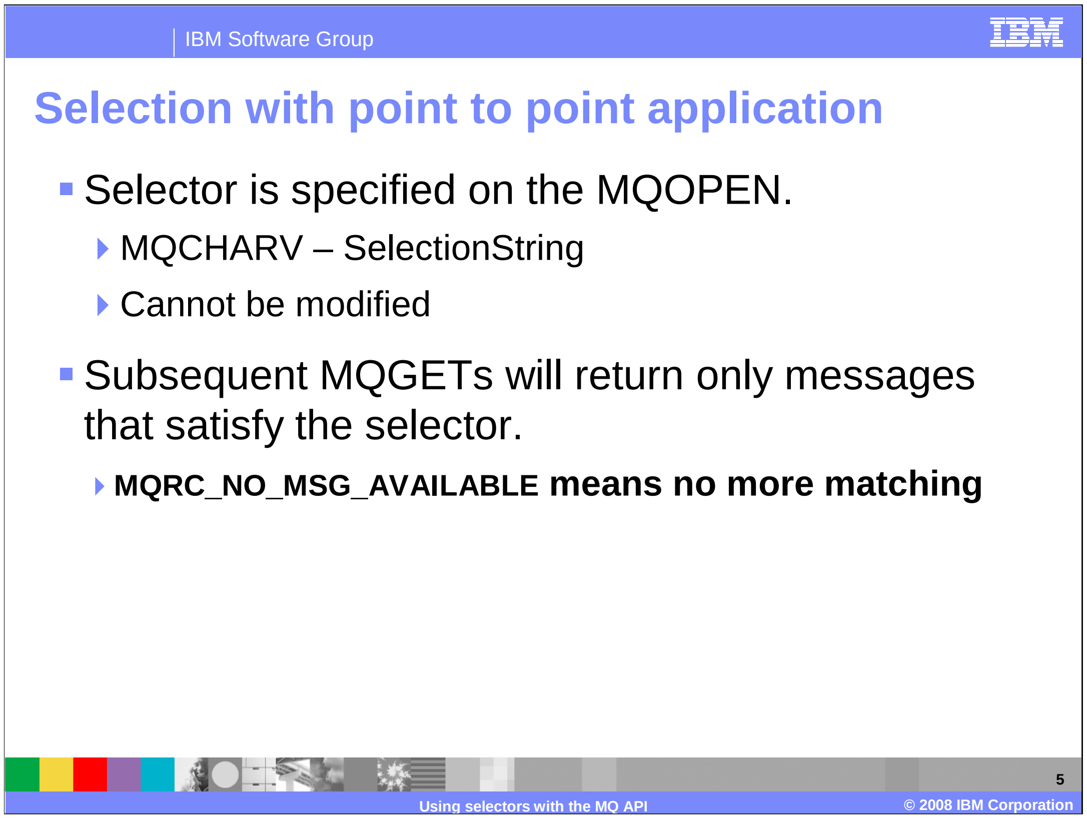
Selectors can use simple string literals testing not only for equality, but also for substring inclusion and others shown on the slide. Byte strings can also be represented in hexadecimal as shown here.

Using numerical properties allows the use of standard arithmetic comparisons, such as greater than as shown here.

Boolean literals TRUE and FALSE allow logical constructs. The full availability of brackets allow composite expressions to be formed. In the example you will get messages relating to cars, over 2500 kilos in weight and blue or green in color. The information center contains detailed reference on how to construct SQL92 selectors.

A message selector is a concept that has been in the JMS specification for a long time. It is a way of limiting the messages that are passed to an application to those that meet certain criteria. Those criteria are based on the values of the message properties and only the value of the message properties. It is important to understand that selection cannot be based on any values of the message payload, only on the message property values. The form of the selector is a string that contains a selection expression. The selection expression is based on the well known SQL92 specification for database queries. The expression will contain references to message properties and constants. The example on the slide shows a selection string that limits messages to those that have a message property name “sport” **AND** where the value of that property is football.

The next slide illustrates some possible expressions that can be used for selection.



Selection with point to point application

- Selector is specified on the MQOPEN.
  - MQCHARV – SelectionString
  - Cannot be modified
- Subsequent MQGETs will return only messages that satisfy the selector.
  - **MQRC_NO_MSG_AVAILABLE means no more matching**

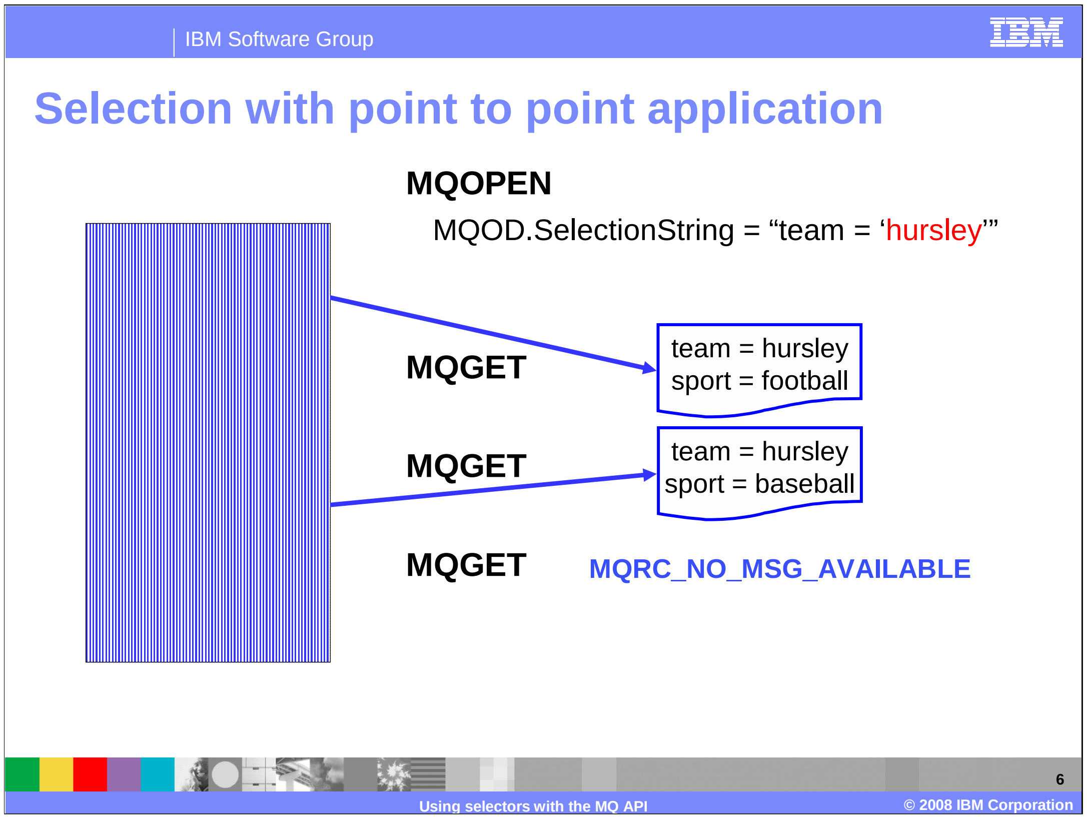
When using message selection in a point to point application (that is – one where messages are being read from a queue) the selection string is supplied as part of the object descriptor (MQOD) on the open call to the queue. It is not permitted on the MQOPEN of a topic.

The SelectionString field in the MQOD is one of the new varying length character strings of type MQCHARV. This means that the string can be of up to ten thousand characters in length. The MQCHARV data type allows the user to specify the Coded Character set ID (CCSID) of the selection string. Care must be taken to ensure that the CCSID is one understood by the queue manager and that property names are matched correctly.

Because the selection string is specified on the MQOPEN it cannot be modified between MQGETs, the same selection string applies to all gets made using the object handle returned from the MQOPEN

MQGETs using this handle will now return only messages that match the selection criteria. This applies to both destructive MQGETs and those with the browse option.

The next slide attempts to illustrate this.



IBM Software Group

Selection with point to point application

MQOPEN

MQOD.SelectionString = “team = ‘hursley’”

Here a queue containing four messages is used as input. The queue contains messages about sports results for various sports and for various teams.

The getting application uses a selection string on the MQOPEN to restrict the messages returned. The selection string asserts that messages must have the team property, and that its value must be hursley. (Home of MQ development) Nothing is said about any other properties.

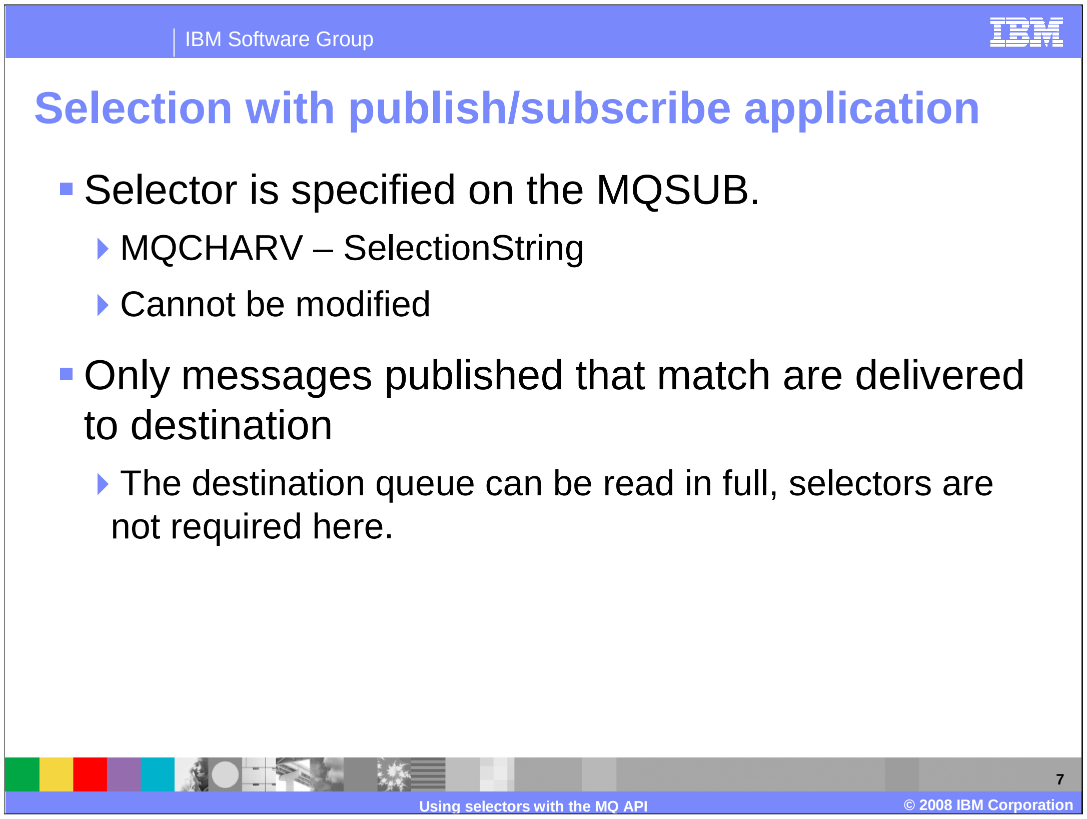
The first MQGET returns the first message from the queue.

The second MQGET returns the **third** message from the queue, omitting message two which, while it had the team property, did not have the value of hursley for the property.

The third MQGET fails with reason code **MQRC_NO_MSG_AVAILABLE,** although messages are still on the queue, none of them have the required property value.

Assuming these were destructive gets and that the application completed normally the queue would then be left with the two unread messages on it.

Next, this presentation will consider use of selectors in a Pub/Sub application.


IBM Software Group

Selection with publish/subscribe application

- Selector is specified on the MQSUB.
  - MQCHARV – SelectionString
  - Cannot be modified
- Only messages published that match are delivered to destination
  - The destination queue can be read in full, selectors are not required here.

7

Using selectors with the MQ API

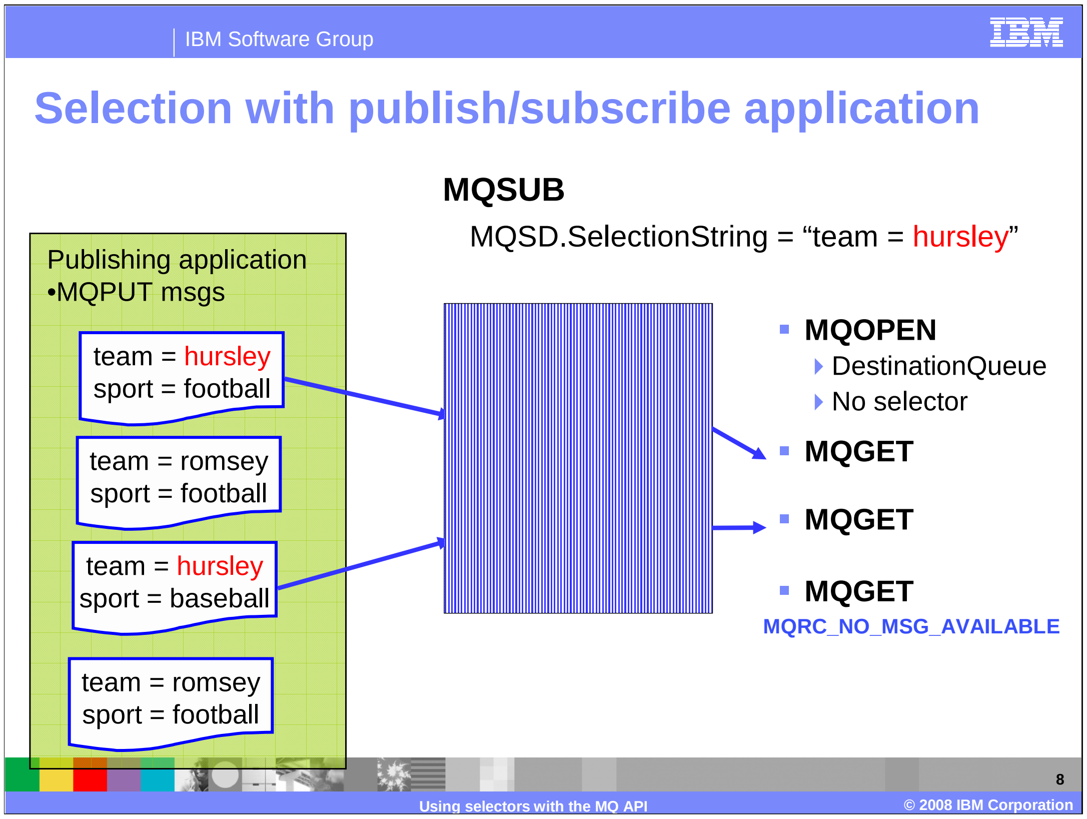
© 2008 IBM Corporation

When using message selection in a pub/sub application the specification is associated with the subscription and is specified in the MQSUB call.

The SelectionString field is this time in the MQSD and is as before of type MQCHARV and cannot be modified, even when a subscription is being altered.

When a message is published to a topic, the queue manager will deliver a copy of the message to every subscriber that subscribes to that topic; but only if the message properties associated with the message matches any selection string associated with the subscription.

The next slide attempts to illustrate this.



IBM Software Group

Selection with publish/subscribe application

MQSUB

MQSD.SelectionString = “team = hursley”

- **MQOPEN**
  - DestinationQueue
  - No selector
- **MQGET**
- **MQGET**
- **MQGET**

MQRC_NO_MSG_AVAILABLE

8

Using selectors with the MQ API

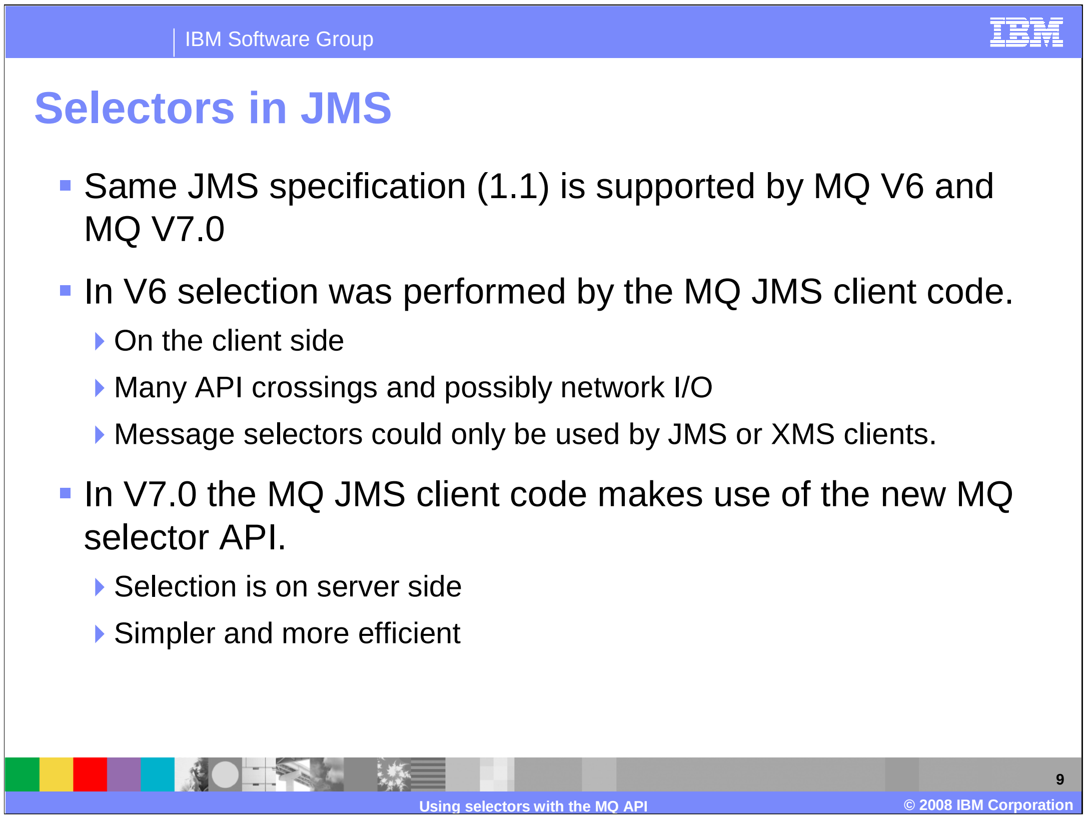
© 2008 IBM Corporation

Here a subscribing application has issued a MQSUB call to set up a subscription to a topic with a selection string saying that only messages with the property team having the value hursley are to be delivered to the destination.

Subsequent to the subscription being established a publishing application starts MQPUTing messages to the topic.

The slide shows that as the messages are put to the topic only those (two) messages meeting the subscription selection are put on the destination queue. So here although the selection is specified at subscription time the queue manager will examine each message being put to a matching topic and will only write those messages that meet the selection criteria to the destination queue.

This means that when the application opens the destination queue named in the subscription to process the messages it can read all the messages – no selection on the MQOPEN for the queue.


IBM Software Group

Selectors in JMS

- Same JMS specification (1.1) is supported by MQ V6 and MQ V7.0
- In V6 selection was performed by the MQ JMS client code.
  - On the client side
  - Many API crossings and possibly network I/O
  - Message selectors could only be used by JMS or XMS clients.
- In V7.0 the MQ JMS client code makes use of the new MQ selector API.
  - Selection is on server side
  - Simpler and more efficient

9

Using selectors with the MQ API

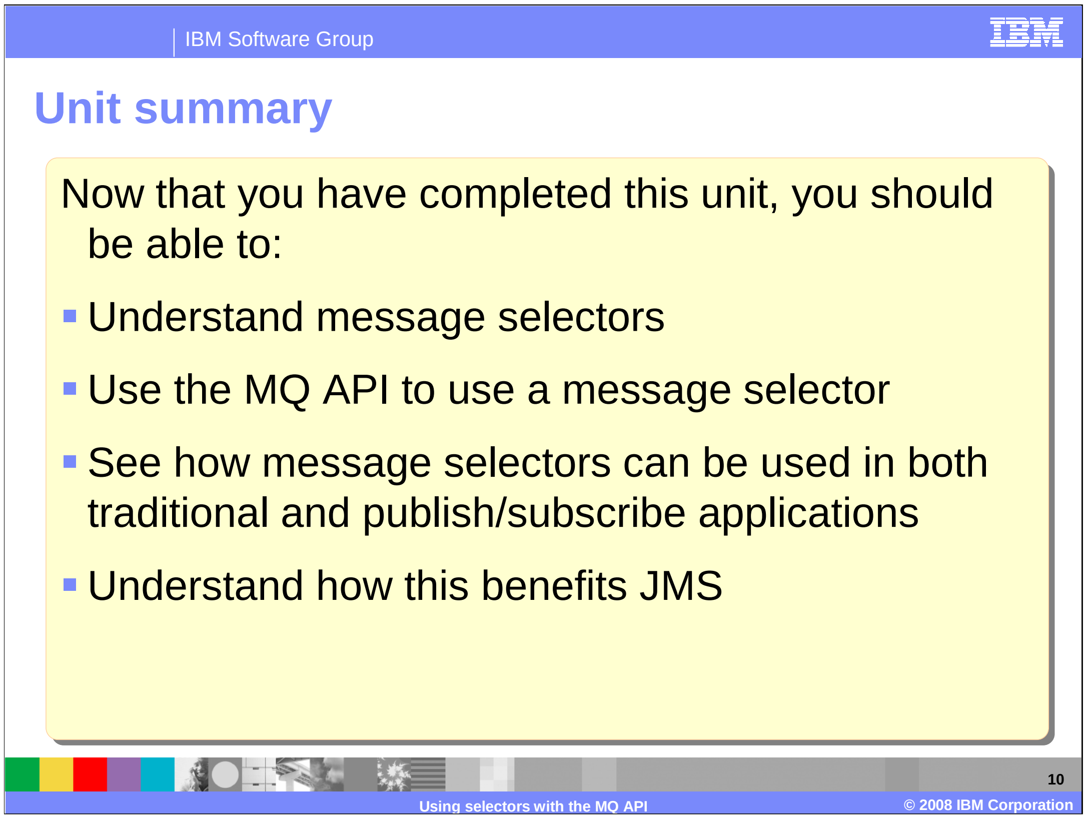
© 2008 IBM Corporation

This slide has two points to make.

The first is the fact that MQ V7.0 supports exactly the same JMS specification as MQ V6 did. This means that JMS applications will function the same running on MQ V7.0 as on V6. The only potential difference in relation to selectors is in the area of performance.

In V6 all the selection was done by the MQ JMS client code. The client code would fetch each message from MQ server, examine the JMS message properties and decide if it should be returned to the application program. When running remotely over a TCP/IP connection the messages that turn out not to be needed still have to travel across the network.

In V7.0 because the MQ API has the required functionality this work is passed to the server side. In a remote environment only the matched messages need flow over the network. All this leads to a Cleaner, Simpler and more efficient implementation.



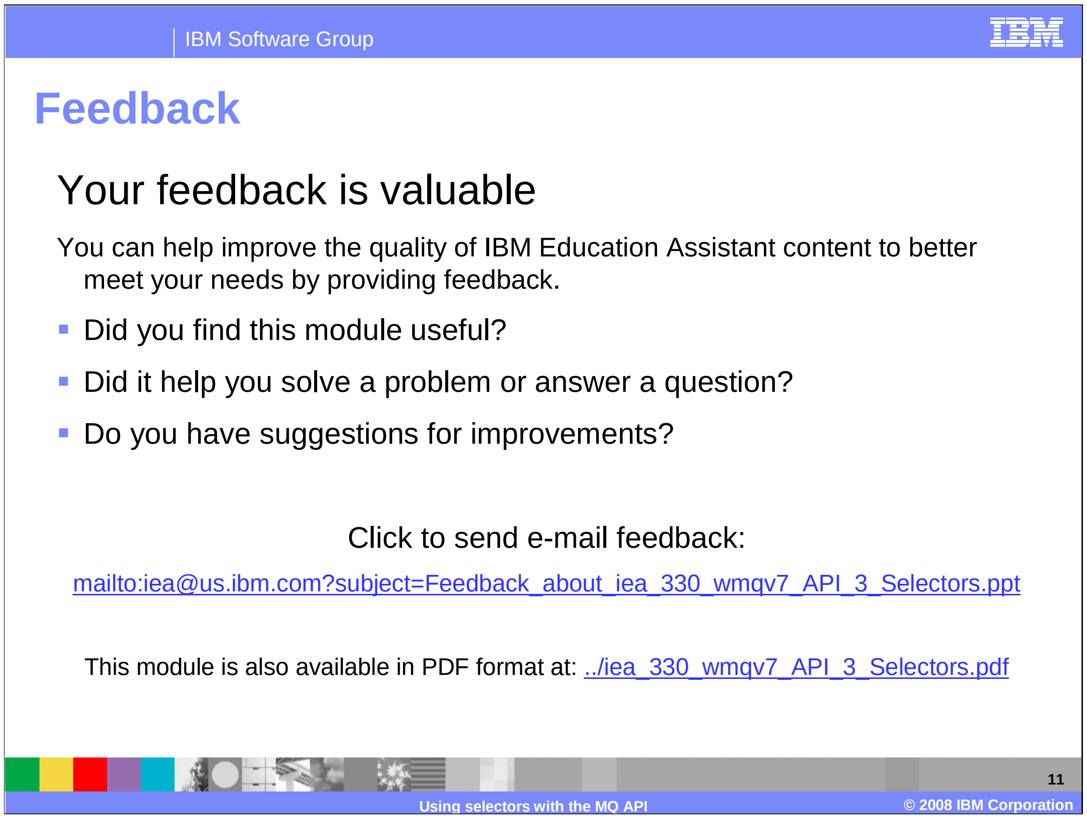
Unit summary

Now that you have completed this unit, you should be able to:

- Understand message selectors
- Use the MQ API to use a message selector
- See how message selectors can be used in both traditional and publish/subscribe applications
- Understand how this benefits JMS

In summary, this presentation has covered what message selectors are and how they are coded in the MQ API, the way they are used in point to point applications as well as pub/sub has been illustrated and the way this simplifies the JMS MQ implementation has also been covered.

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.

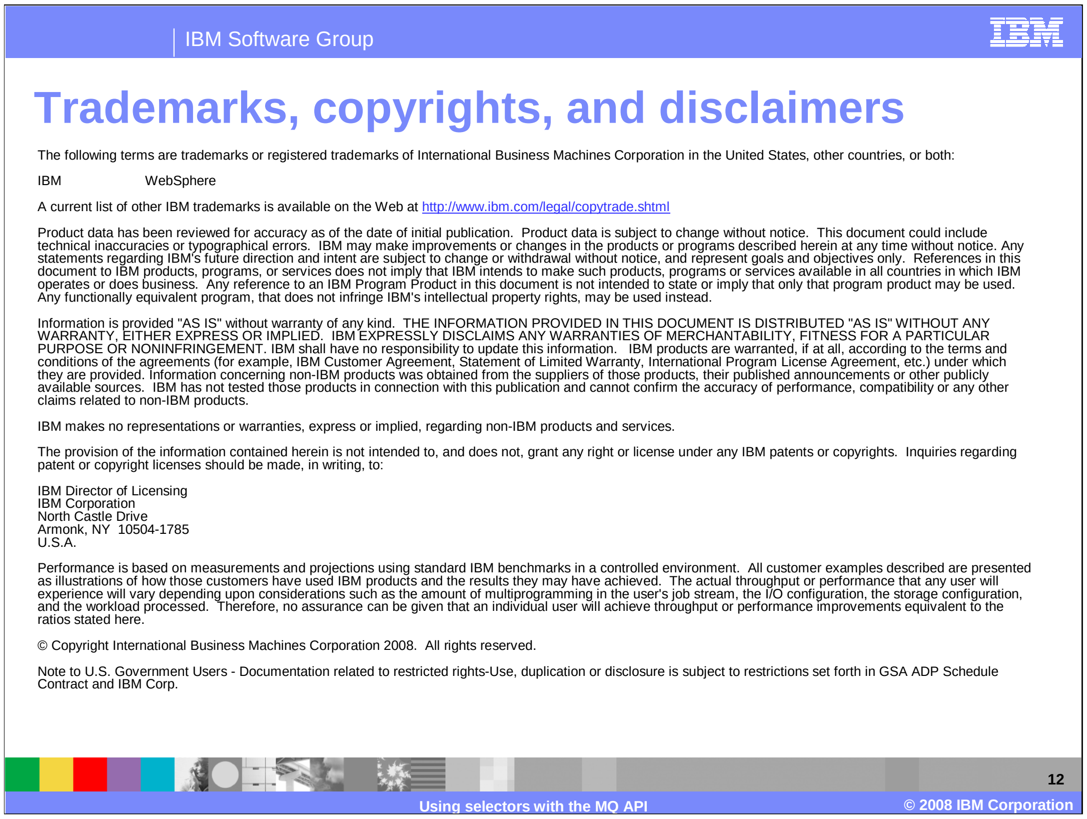
- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_iea_330_wmqv7_API_3_Selectors.ppt

This module is also available in PDF format at: ../iea_330_wmqv7_API_3_Selectors.pdf

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



Trademarks, copyrights, and disclaimers