IBM Software Group

WebSphere® MQ V7.0

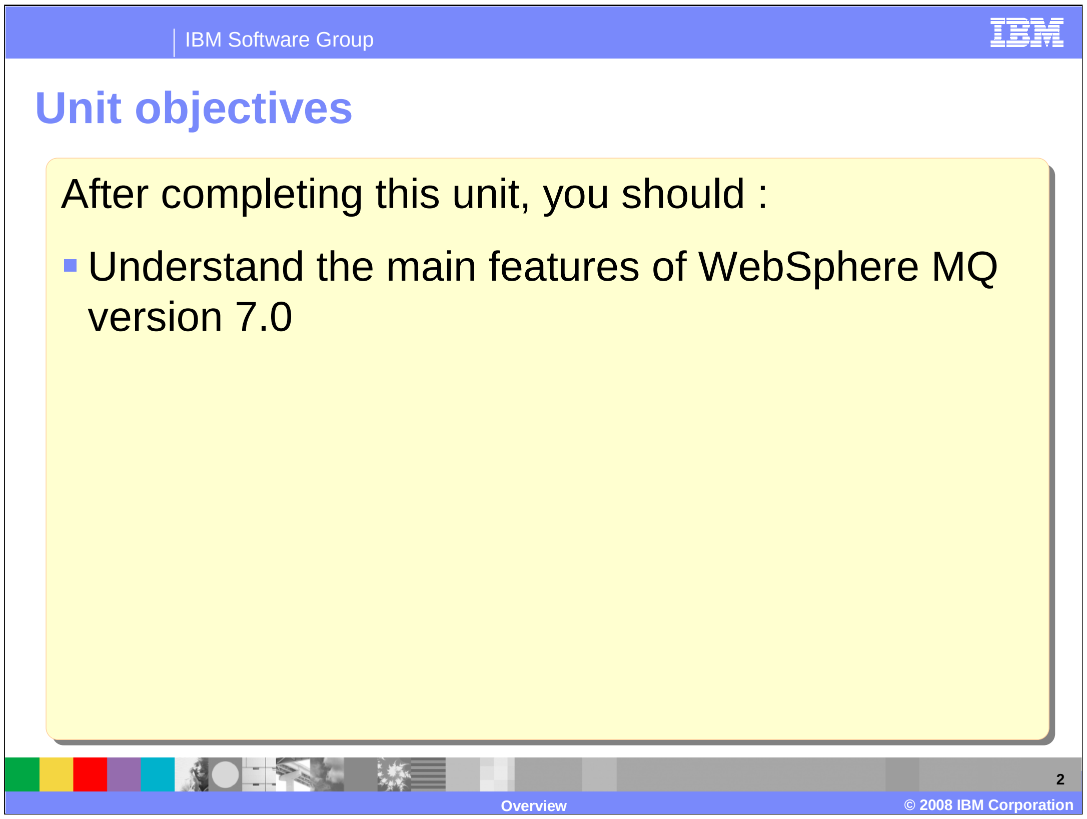
Overview

business on demand.

This unit presents an overview of the new features of WebSphere MQ version 7.0.

All the features references in this presentation are covered in more detail in later presentations.

This unit assumes a reasonable understanding of the existing WebSphere MQ product.



Unit objectives

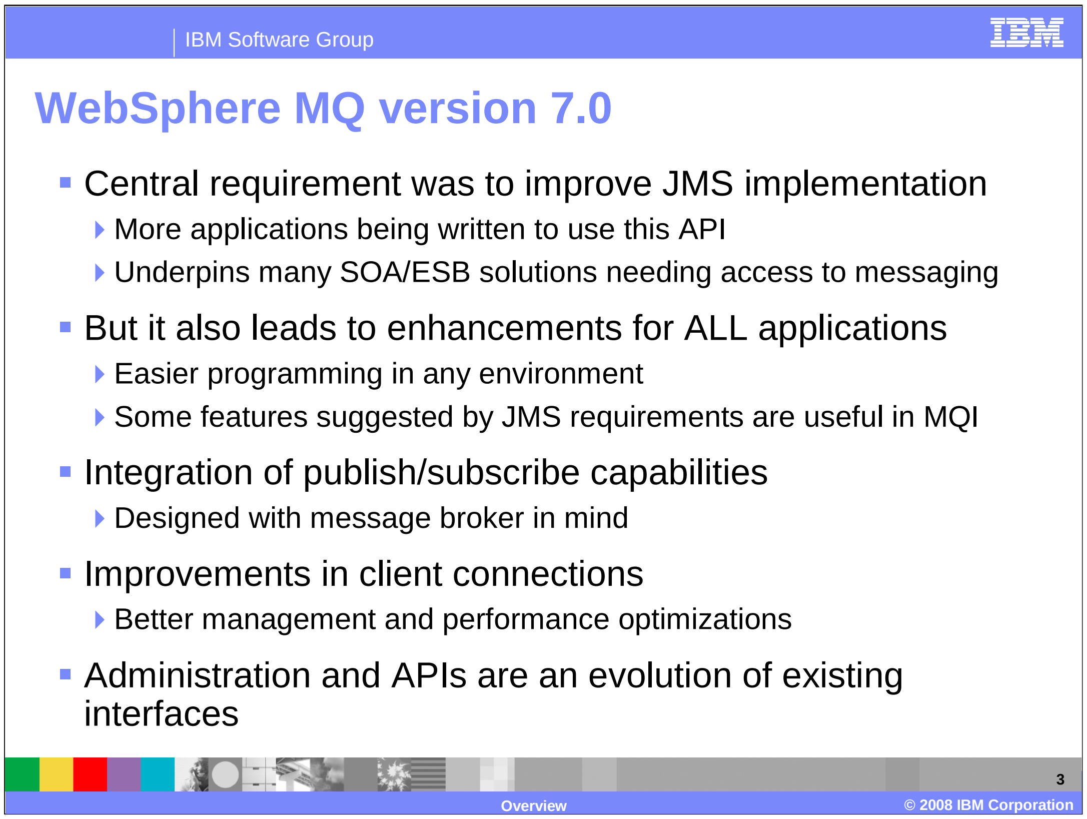
After completing this unit, you should :

- Understand the main features of WebSphere MQ version 7.0

After completing this unit you should have some understanding of the main features of WebSphere MQ version 7.0.

This unit does not attempt to cover the full range of features or how they can be used.

Further information is presented in later units or you should refer to the product information center.



WebSphere MQ version 7.0

- Central requirement was to improve JMS implementation
  - More applications being written to use this API
  - Underpins many SOA/ESB solutions needing access to messaging
- But it also leads to enhancements for ALL applications
  - Easier programming in any environment
  - Some features suggested by JMS requirements are useful in MQI
- Integration of publish/subscribe capabilities
  - Designed with message broker in mind
- Improvements in client connections
  - Better management and performance optimizations
- Administration and APIs are an evolution of existing interfaces

WebSphere MQ Version 7.0 takes its central theme from requirements to extend its support for JMS applications. WebSphere MQ has supported the JMS API for many years. Recent growth in user JMS applications, and growth in SOA solutions that need a JMS provider, have increased the importance of this programming interface.

This has lead to incorporation of several JMS features into the core WebSphere MQ engine. In turn this has allowed these features to be easily accessible to applications using the native MQ API (MQI), not just those written to the JMS API.

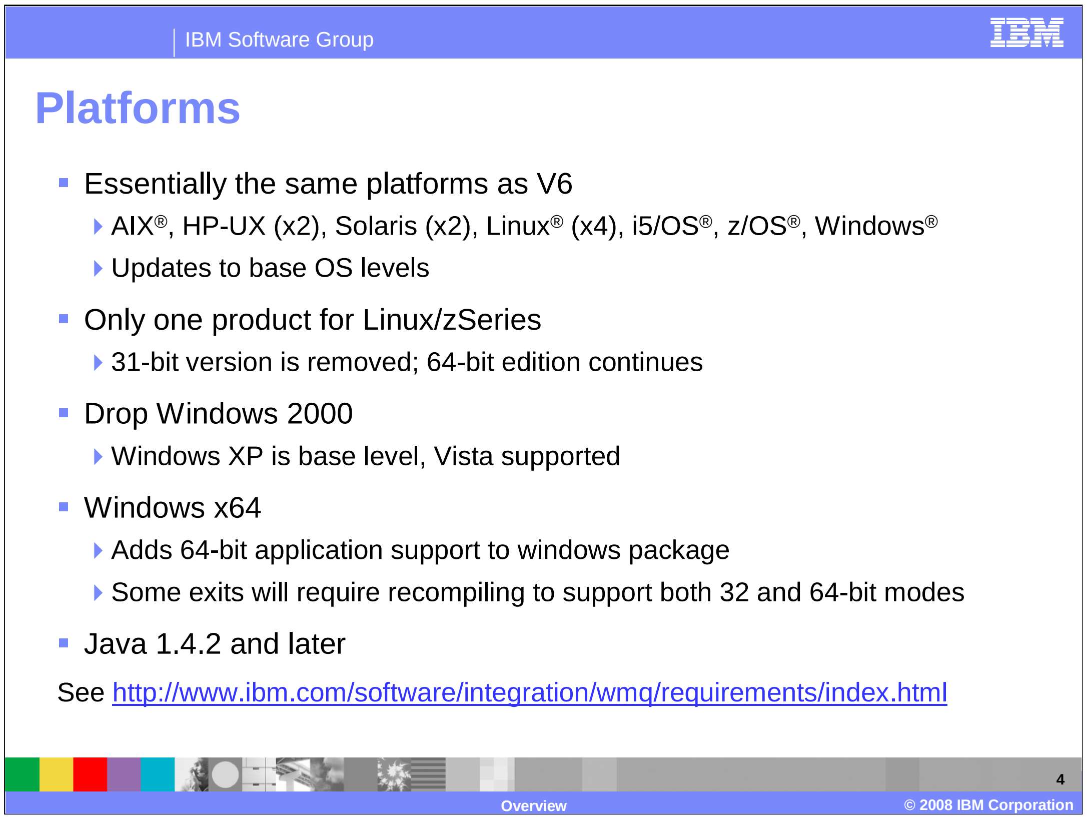
The publish/subscribe messaging pattern is one of the two patterns defined by the JMS API, the other being the point-to-point pattern. WebSphere MQ Version 7.0 provides publish/subscribe capabilities that exploit the natural high availability capabilities in WebSphere MQ including queue manager clustering. It incorporates a comprehensive range of qualities of service for message delivery, ranging from assured delivery to lightweight non persistent. It also permits a range of topologies, including support for very large numbers of subscribers.

The pub/sub implementation in V7.0 has also been designed to ensure that a simplified, and fully-integrated, relationship with Message Broker's pub/sub capabilities can be achieved.

WebSphere MQ version 7.0 provides application programming interfaces that support the concepts required by JMS, including publish/subscribe, in the MQI. These base MQ APIs are then exploited by the JMS implementation.

Enhancements to the programming and administrative interfaces are all modeled after existing WebSphere MQ interfaces, to deliver a familiar and easy-to-learn route towards the new features.

This presentation is not an exhaustive list of all of the new features of WebSphere MQ version 7, but gives an overview of major functions.



Platforms

- Essentially the same platforms as V6
  - AIX®, HP-UX (x2), Solaris (x2), Linux® (x4), i5/OS®, z/OS®, Windows®
  - Updates to base OS levels
- Only one product for Linux/zSeries
  - 31-bit version is removed; 64-bit edition continues
- Drop Windows 2000
  - Windows XP is base level, Vista supported
- Windows x64
  - Adds 64-bit application support to windows package
  - Some exits will require recompiling to support both 32 and 64-bit modes
- Java 1.4.2 and later

See http://www.ibm.com/software/integration/wmq/requirements/index.html

The set of supported platforms is essentially the same as in the previous version of WebSphere MQ. There are changes to the base levels of operating system that are supported, driven by the current status of each OS. V7.0 ships 12 different versions, to handle the different operating system and hardware combinations.

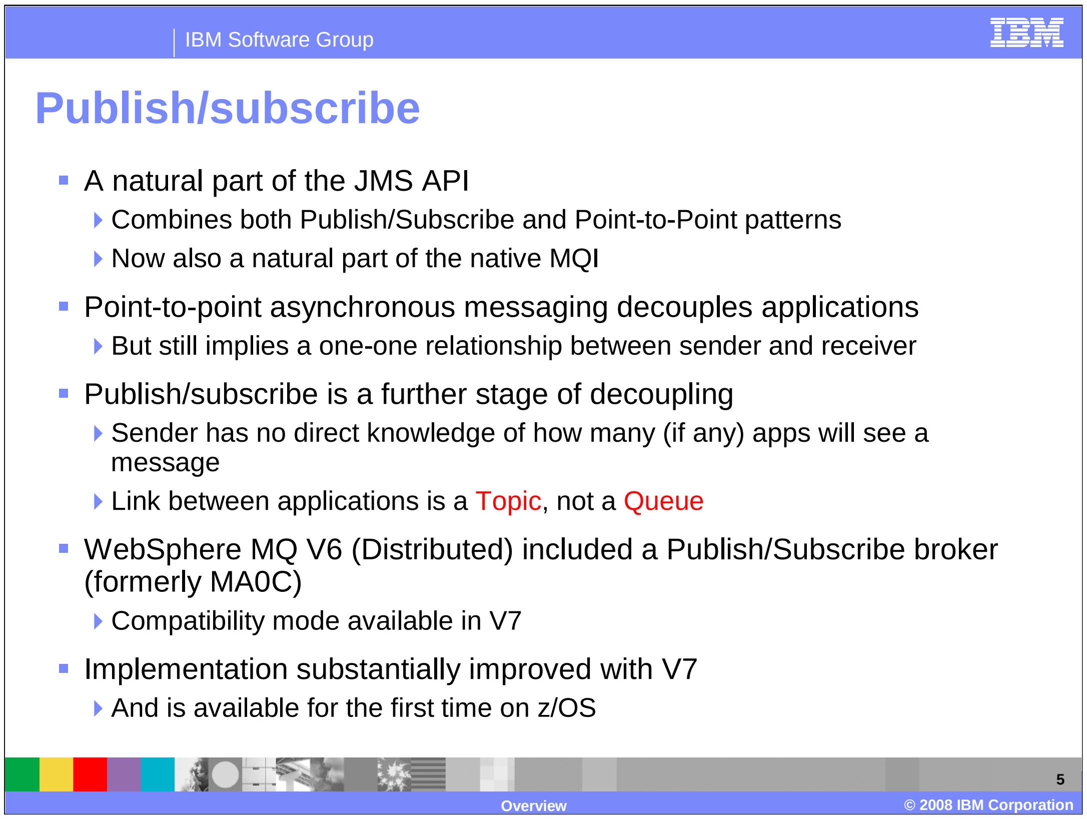
In V6 there were two versions of WebSphere MQ for Linux/zSeries, one supporting only 31-bit applications and a second that incorporated 64-bit support. For V7.0 the first of these has been dropped, but the 64-bit version is fully compatible with 31-bit applications.

One of the changes to Windows support is that Windows 2000 is no longer a supported platform. Windows XP is the oldest level that can be used. With this release support is included for Vista. There is also support for the x64 versions of the operating system. When running on a 64-bit platform, the queue manager itself is still a 32-bit process, but support is now included for 64-bit applications to use this queue manager. Exits that run inside application code, namely API Exits and Data Conversion Exits, will need to have both 32-bit and 64-bit versions available. 64-bit clients   need 64-bit channel exits as well. There is only one version of WebSphere MQ for Windows; it incorporates support for all the variations of the operating system.

Java 1.4.2 is the lowest level of runtime supported; later levels are also supported

For z/OS the prerequisite level is 1.8. (z/OS 1.7 has its end of service set to September 2008.

For most recent information on the SOE (Supported Operating Environments), always look at the Web page. That is kept updated in between releases of the product.


IBM Software Group

Publish/subscribe

- A natural part of the JMS API
  - Combines both Publish/Subscribe and Point-to-Point patterns
  - Now also a natural part of the native MQI
- Point-to-point asynchronous messaging decouples applications
  - But still implies a one-one relationship between sender and receiver
- Publish/subscribe is a further stage of decoupling
  - Sender has no direct knowledge of how many (if any) apps will see a message
  - Link between applications is a Topic, not a Queue
- WebSphere MQ V6 (Distributed) included a Publish/Subscribe broker (formerly MA0C)
  - Compatibility mode available in V7
- Implementation substantially improved with V7
  - And is available for the first time on z/OS

5

Overview

© 2008 IBM Corporation

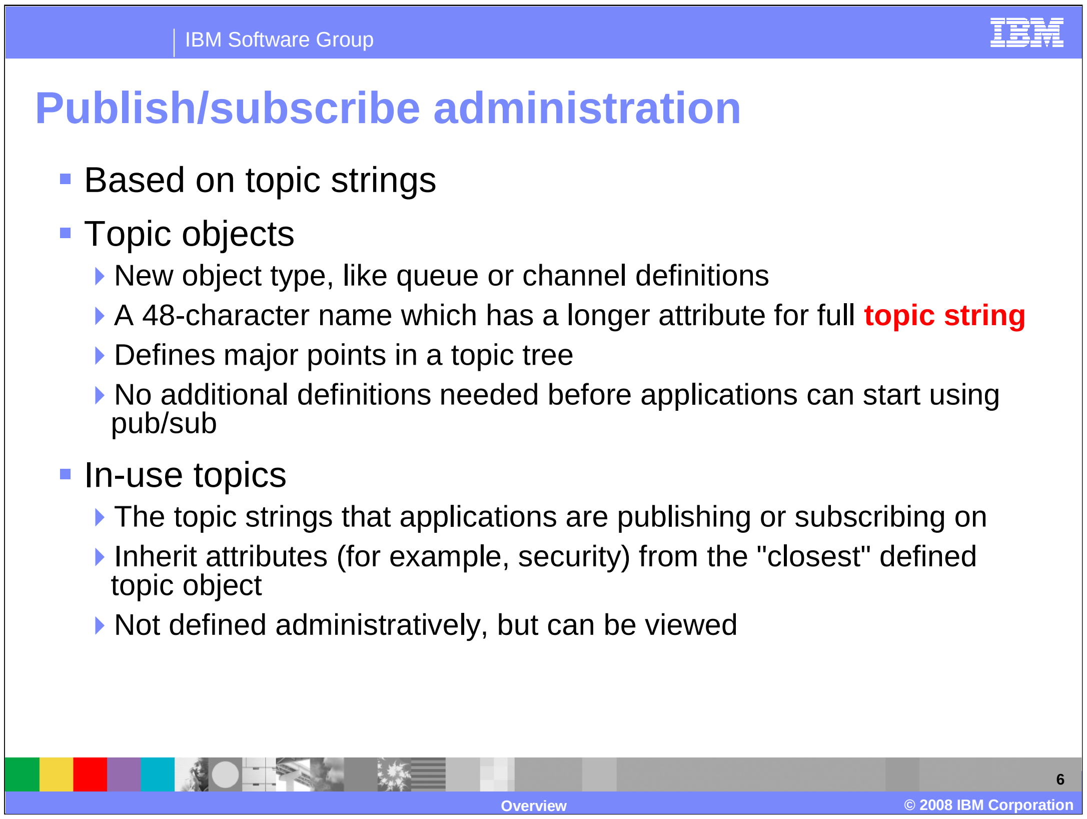
Point-to-point asynchronous messaging decouples applications from each other but still implies a one-one relationship between sender and receiver. Publish/subscribe is a further stage of decoupling since the sender has no direct knowledge of how many (if any) applications will see a message. In Publish/subscribe, the link between applications is a topic, not a queue. In the JMS API, both topics and queues are referred to as destinations.

Publish/subscribe is a natural part of the JMS API and with WebSphere MQ version 7.0 it is also a natural part of the native MQI. New verbs and options, that are shown later, make it easy to use pub/sub from any environment.

In WebSphere MQ version 7.0 there is a publish/subscribe engine as part of the queue manager on all distributed and z/OS systems. This is addressed directly through the MQI, replacing the queue-based interface available in WebSphere MQ V6, and previously as the MA0C SupportPac®. The administration model for this uses the standard WebSphere MQ model, replacing the interfaces available in WebSphere MQ V6. Although these older interfaces are now deprecated, they are still supported to ensure coexistence with earlier versions of applications that connect to a version 7.0 queue manager.

This publish/subscribe engine is part of the queue manager and is always available. It does not have to be manually started. A queue-based publish/subscribe daemon task is provided to ensure coexistence of the V6 style publish/subscribe with V7.0 and to migrate the use of the V6 interface to V7.

WebSphere MQ on z/OS has not previously had a native pub/sub capability, and has had to rely on Message Broker or connecting to WebSphere MQ on a distributed platform for access to pub/sub services. That limitation is removed in V7.



Publish/subscribe administration

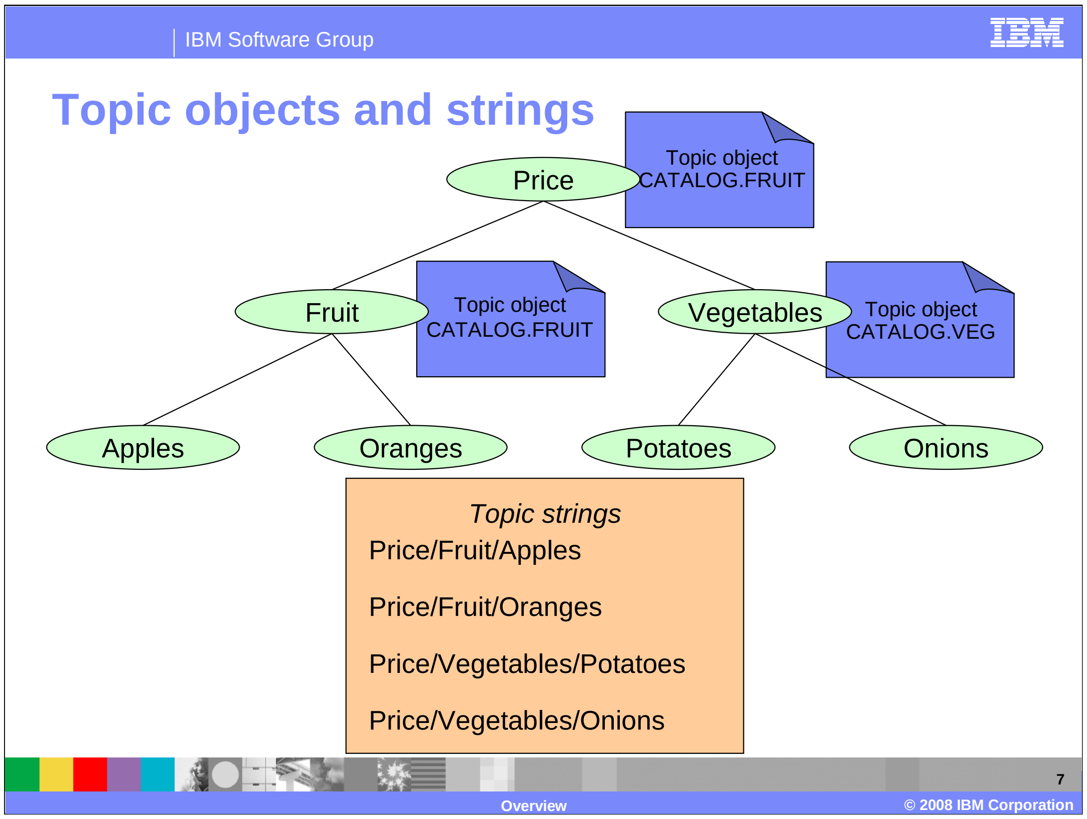
- Based on topic strings
- Topic objects
  - New object type, like queue or channel definitions
  - A 48-character name which has a longer attribute for full **topic string**
  - Defines major points in a topic tree
  - No additional definitions needed before applications can start using pub/sub
- In-use topics
  - The topic strings that applications are publishing or subscribing on
  - Inherit attributes (for example, security) from the "closest" defined topic object
  - Not defined administratively, but can be viewed

The construct that couples a publishing application with subscribing applications is the topic string. A topic string can be very long, and has a structure implied by the separator character forward slash '/'. Topic strings can be used directly from publishing and subscribing applications without any need to define them administratively.

However, there can sometimes be a need to tune certain attributes of topic strings or to apply security to topic strings and as such an administrative point is required in order to do this. This administrative point is a new object type called a topic object. It is envisaged that the top layer, or the top few layers, of a topic string hierarchy is defined administratively. Here any appropriate attributes, such as security, can be set. The remaining layers of the hierarchy are defined by being used, inheriting attributes from their parents in the hierarchy.

Managing topics administratively is done by means of commands in MQSC or PCF as per the standard WebSphere MQ administration model allowing creation, alteration, deletion and display of topic objects (which contain topic string).

In either case, whether a topic string is defined administratively or automatically created when an application uses it, status can be interrogated from the queue manager. Similarly, statistics can be gathered on the use of a topic string regardless of its creation mechanism.



Topic objects and strings

Topic strings

Price/Fruit/Apples

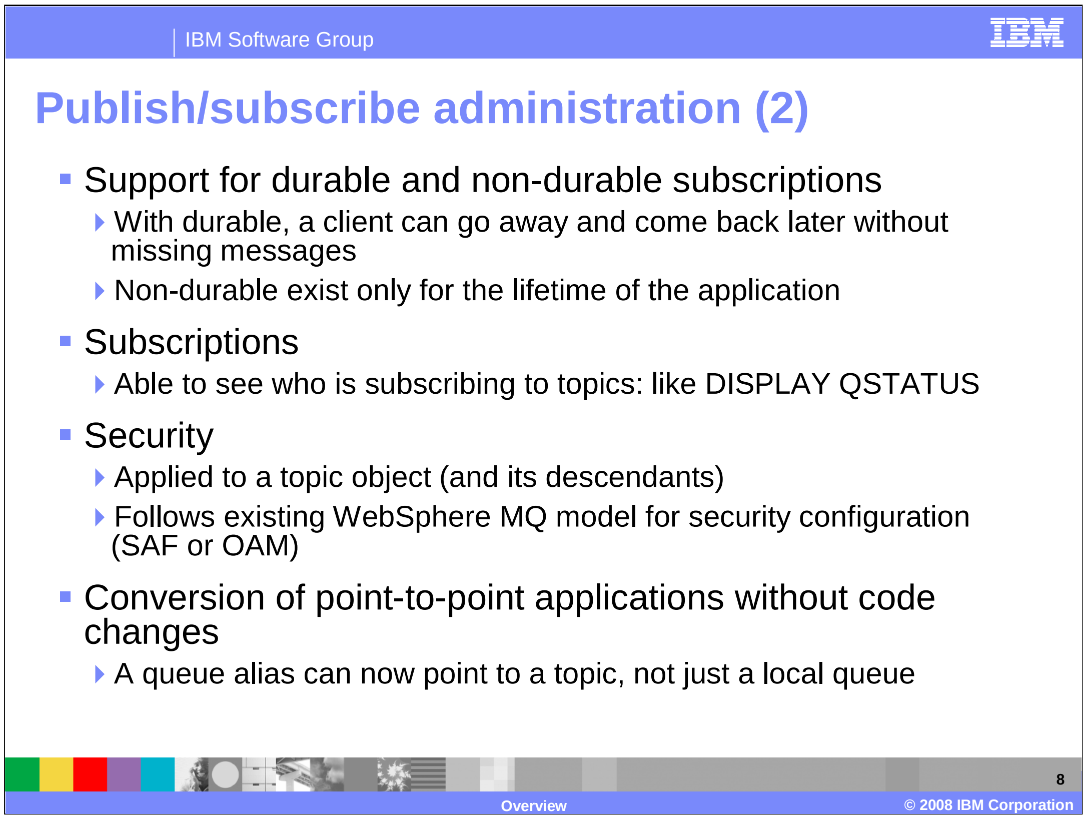
Price/Fruit/Oranges

Price/Vegetables/Potatoes

Price/Vegetables/Onions

Topic strings can be any characters you choose. You can, and should, add structure to you topic strings using the '/' character. This produces a topic tree with a hierarchical structure, as shown here. This hierarchical topic tree was created by the use of the topic strings shown; however, it is generally pictured as a tree.

This slide illustrates a possible structuring. The topic strings representing the pricing of various fruits and vegetables form naturally into a tree structure. At appropriate points in the hierarchy WebSphere MQ topic objects are created. These topic objects can have security and other attributes set administratively. These attributes will then be inherited by its children.



Publish/subscribe administration (2)

- Support for durable and non-durable subscriptions
  - With durable, a client can go away and come back later without missing messages
  - Non-durable exist only for the lifetime of the application
- Subscriptions
  - Able to see who is subscribing to topics: like DISPLAY QSTATUS
- Security
  - Applied to a topic object (and its descendants)
  - Follows existing WebSphere MQ model for security configuration (SAF or OAM)
- Conversion of point-to-point applications without code changes
  - A queue alias can now point to a topic, not just a local queue

Durable subscriptions can continue collecting publications after the starting task completes. This means that no messages are missed, but can lead to queues becoming full. Non durable subscriptions, by contrast, terminate with the task that starts them. The queue manager can do all required cleanup when a non durable subscription terminates.

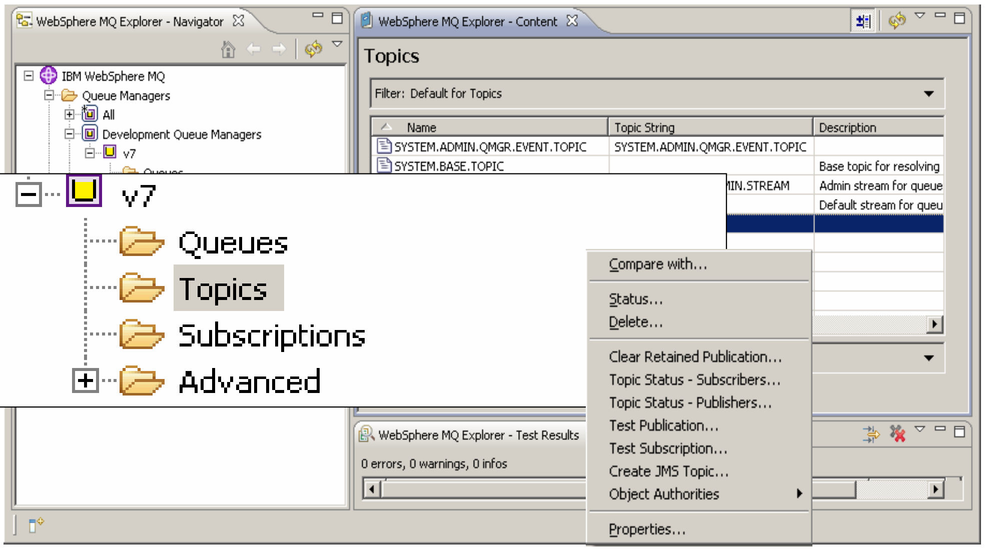
A range of administrative functions are available for subscriptions. Allow administrator to display who is subscribing to what topics and the number of messages delivered.

The security of who can use a particular topic string is something that is restricted by permissions on the associated topic object. In the same way that attributes can be inherited from parent topics in a topic hierarchy, security settings can also be inherited from parent topics. It follows the existing WebSphere MQ model for security configuration (SAF on z/OS and the OAM on distributed). The z/OS security interface has been enhanced to permit use of mixed-case support in RACF to give more flexibility in the naming of topic objects.

Topic security is checked for publishing applications before they are allowed to publish, and subscribing applications before they are allowed to subscribe to a topic. Different permissions exist for creation of a new subscription against resumption of an existing durable subscription.

Applications which currently put to or get from a queue and cannot have their source code changed, can still publish messages through an alias queue which references a topic object.

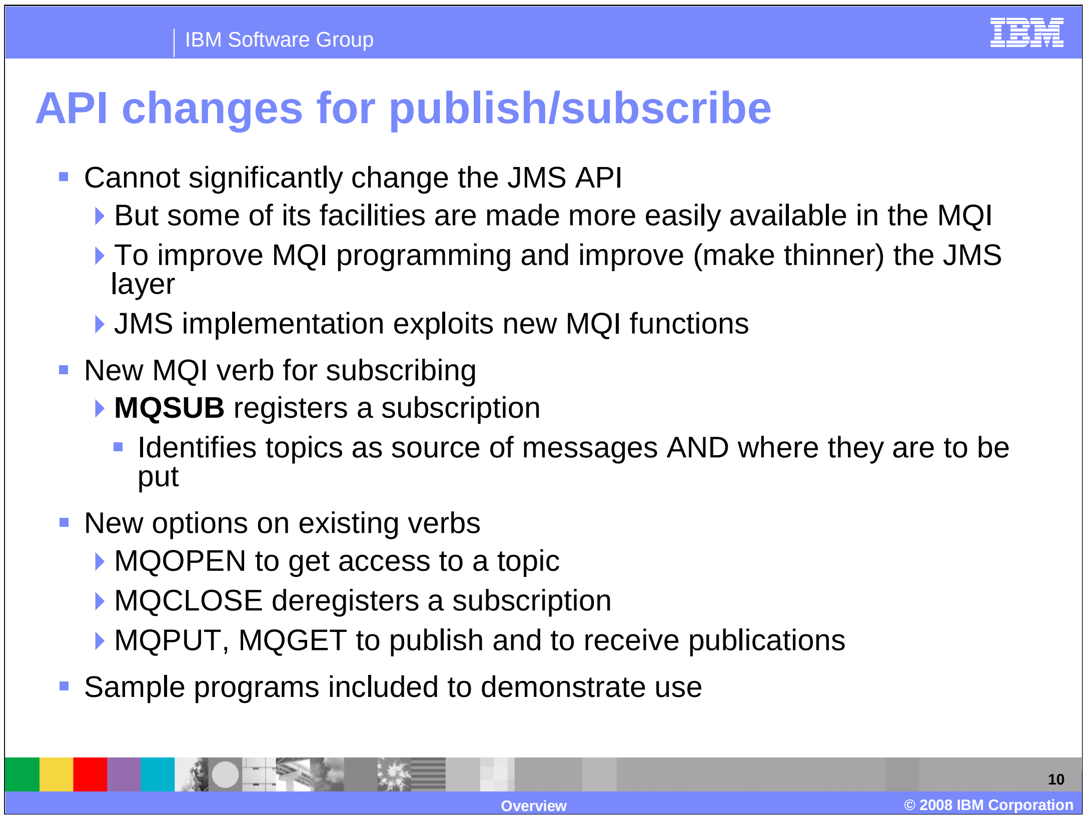
Publish/subscribe in the MQ explorer


WebSphere MQ Explorer - Navigator
IBM WebSphere MQ
Queue Managers
All
Development Queue Managers
v7
v7
Queues
Topics
Subscriptions
Advanced
WebSphere MQ Explorer - Content
Topics
Filter: Default for Topics
Compare with...
Status...
Delete...
Clear Retained Publication...
Topic Status - Subscribers...
Topic Status - Publishers...
Test Publication...
Test Subscription...
Create JMS Topic...
Object Authorities
Properties...
WebSphere MQ Explorer - Test Results
0 errors, 0 warnings, 0 infos

Name	Topic String	Description
SYSTEM.ADMIN.QMGR.EVENT.TOPIC	SYSTEM.ADMIN.QMGR.EVENT.TOPIC	
SYSTEM.BASE.TOPIC		Base topic for resolving
	IN.STREAM	Admin stream for queue
		Default stream for queu

Topics and subscriptions are supported from MQ Explorer as a new category of objects like QUEUES with a similar set of operations. This includes the ability to perform a test publish and create a test subscription.

The various administrative options are covered in a later presentation.


IBM Software Group

API changes for publish/subscribe

- Cannot significantly change the JMS API
  - But some of its facilities are made more easily available in the MQI
  - To improve MQI programming and improve (make thinner) the JMS layer
  - JMS implementation exploits new MQI functions
- New MQI verb for subscribing
  - **MQSUB** registers a subscription
    - Identifies topics as source of messages AND where they are to be put
- New options on existing verbs
  - MQOPEN to get access to a topic
  - MQCLOSE deregisters a subscription
  - MQPUT, MQGET to publish and to receive publications
- Sample programs included to demonstrate use

10

Overview © 2008 IBM Corporation

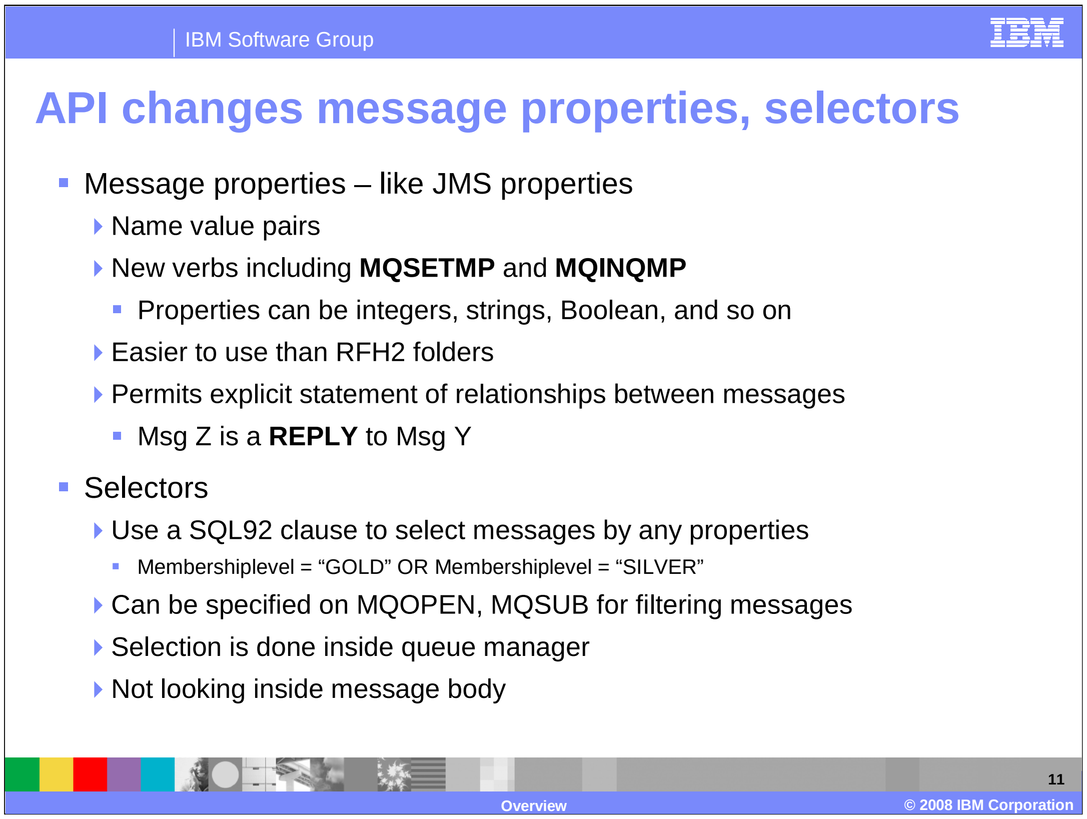
Publish/subscribe is part of the JMS API. MQ cannot change the JMS API but version 7.0 makes some of its facilities directly available in the MQI. This has the effect of improving MQI programming and shrinks the JMS layer that is built on the MQI.

Publishing to a topic and subscribing for publications on a topic are now part of the MQI. Subscribing to a topic object registers that applications interest in publications to that topic. This is done by means of a new verb, MQSUB. Reading from the returned object handle gives the application any publications made to that topic.

Opening a topic for output registers that application as a publisher to that topic for the duration of the open. Publishing is done by using MQPUT to put messages to the returned object handle. This sends them to all the subscribers of that topic.

MQSUB is the subscription verb, it identifies the topic being subscribed to AND identifies a queue where messages published to that topic are to be put. The topic string can contain wildcards in which case many topic strings are being subscribed to in a single subscription.

A range of sample programs, such as AMQSPUB, are supplied, which illustrate these new functions.



API changes message properties, selectors

- Message properties – like JMS properties
  - Name value pairs
  - New verbs including **MQSETMP** and **MQINQMP**
    - Properties can be integers, strings, Boolean, and so on
  - Easier to use than RFH2 folders
  - Permits explicit statement of relationships between messages
    - Msg Z is a **REPLY** to Msg Y
- Selectors
  - Use a SQL92 clause to select messages by any properties
    - Membershiplevel = “GOLD” OR Membershiplevel = “SILVER”
  - Can be specified on MQOPEN, MQSUB for filtering messages
  - Selection is done inside queue manager
  - Not looking inside message body

Message properties are arbitrary name value pairs that can be associated with a message. They are not part of the message body but act as an extended, and extensible, message descriptor. They are better encapsulated and easier to use then RFH2 folders previously used for this. The properties can be used to express an explicit relationship between messages. Such as this message is a reply to this message. All existing MQMD properties can be exposed as message properties and the message properties can be accessed by JMS Applications as properties in the JMS sense.

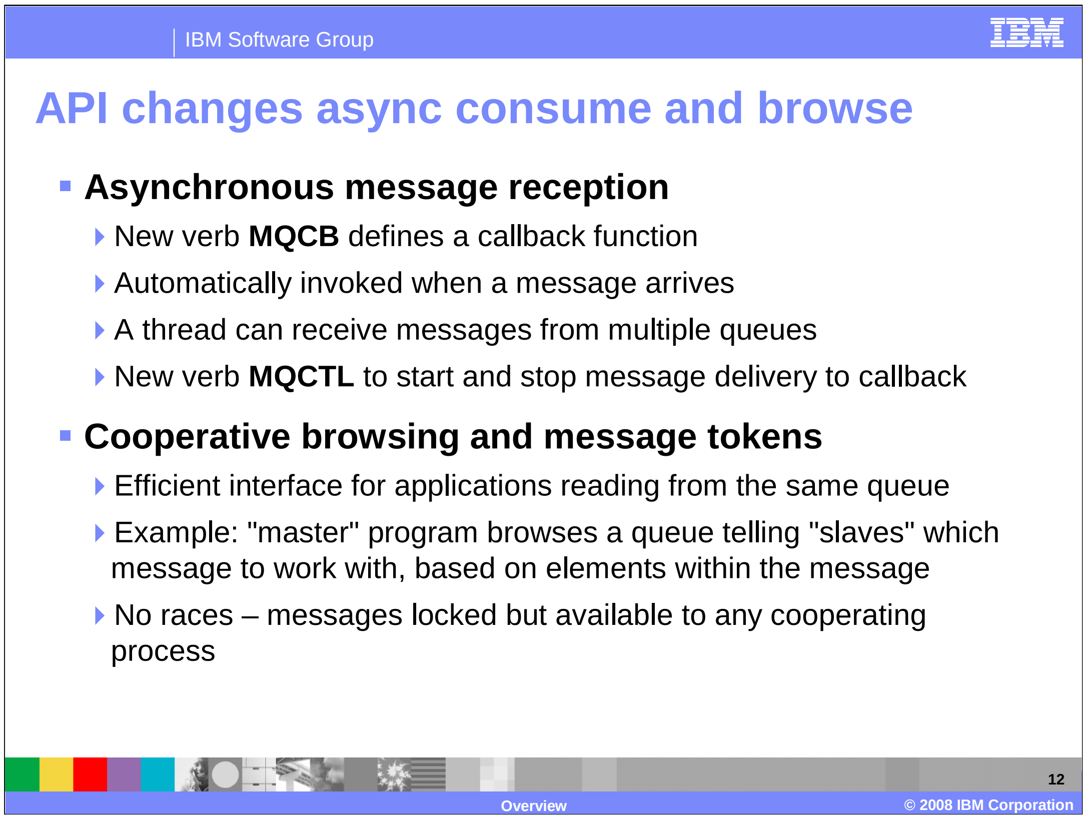
Selectors are another JMS concept now delivered natively in the MQ API.

A selector string is an expression that uses any of the message properties to limit the messages that are passed to an application. They can be used on the MQOPEN for a queue and on an MQSUB when subscribing to a topic.

For example, an application might only need to read messages where the message property “Membershiplevel” has been set to either GOLD or SILVER.

Since these selectors are understood by the queue manager, the JMS implementation has been made simpler and more efficient than previously.

Selection can be made based on the message properties, (including the MQMD fields), only NOT on the content of the message body. WebSphere Message Broker is still required for filtering based on the content, or body, of the message.



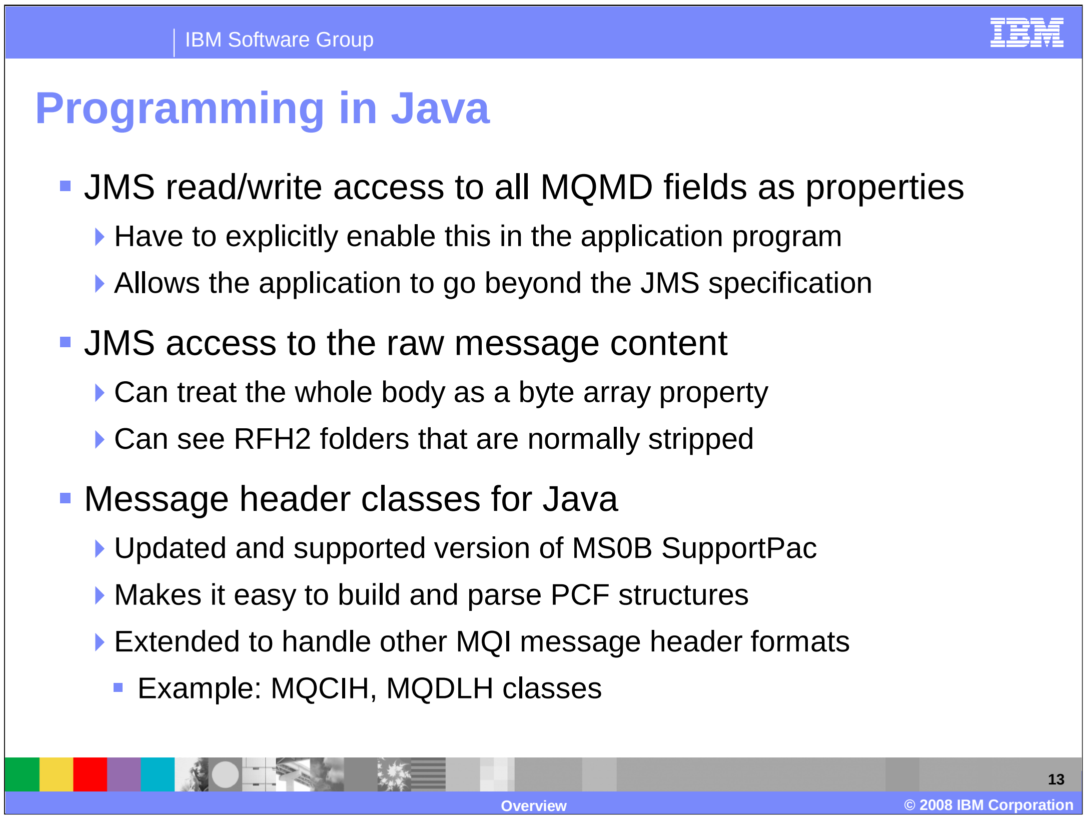
API changes async consume and browse

- **Asynchronous message reception**
  - New verb **MQCB** defines a callback function
  - Automatically invoked when a message arrives
  - A thread can receive messages from multiple queues
  - New verb **MQCTL** to start and stop message delivery to callback
- **Cooperative browsing and message tokens**
  - Efficient interface for applications reading from the same queue
  - Example: "master" program browses a queue telling "slaves" which message to work with, based on elements within the message
  - No races – messages locked but available to any cooperating process

The MQI is enhanced to provide asynchronous consumption of messages. This allows applications to be called back when a message becomes available, meeting their message selection criteria on one of their named queues. The callback function is also invoked when something happens that requires the application to end; for example when the connection to the queue manager is being quiesced.

This removes the need for the JMS interface to build the onMessage() method on top of polling MQGET calls and leads to a simpler more efficient implementation.

Message browsing has been enhanced by a mark function and by the use of message tokens. An application, or a group of cooperating applications, can mark a message as having been processed. The mark stops the applications from rereading the message later. The new message token provides a unique reference to a message and allows a reference to a marked message to be passed to another application that will process the message. A major application of this is expected to be dispatcher applications processing a queue of workitems by selecting items from the queue and passing them to an appropriate processing application based on message properties or content.

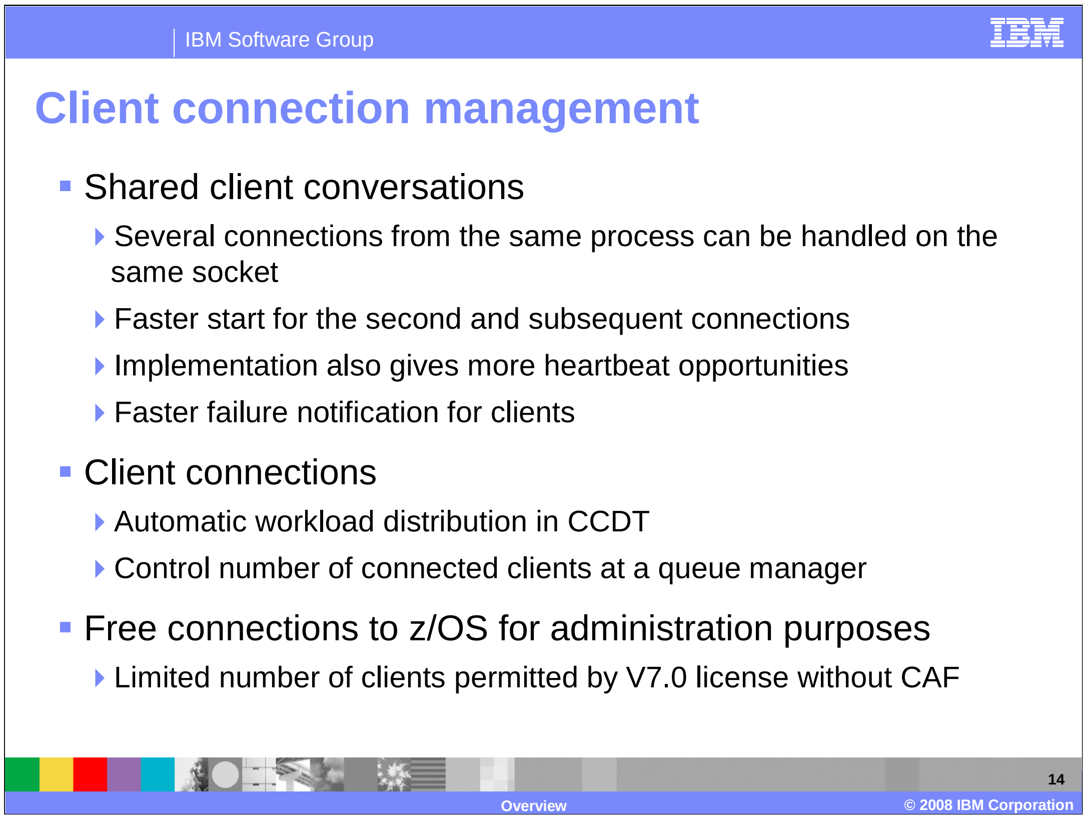


Programming in Java

- JMS read/write access to all MQMD fields as properties
  - Have to explicitly enable this in the application program
  - Allows the application to go beyond the JMS specification
- JMS access to the raw message content
  - Can treat the whole body as a byte array property
  - Can see RFH2 folders that are normally stripped
- Message header classes for Java
  - Updated and supported version of MS0B SupportPac
  - Makes it easy to build and parse PCF structures
  - Extended to handle other MQI message header formats
    - Example: MQCIH, MQDLH classes

WebSphere MQ Version 7.0 does include new interfaces for the Java programmer. One group of these is intended to make it easier to create and parse MQI structures such as the PCF elements. The MS0B SupportPac has been enhanced and incorporated into the product as a fully-supported component.

There are also options to simplify the interoperation of JMS programs with other programming languages, allowing more direct manipulation of fields in the MQMD and giving access to the raw message body. Because modifying some of these attributes is not permitted within the JMS specification, the program has to "acknowledge" that it is prepared to do this by setting another virtual property on the message or destination.



Client connection management

- Shared client conversations
  - Several connections from the same process can be handled on the same socket
  - Faster start for the second and subsequent connections
  - Implementation also gives more heartbeat opportunities
  - Faster failure notification for clients
- Client connections
  - Automatic workload distribution in CCDT
  - Control number of connected clients at a queue manager
- Free connections to z/OS for administration purposes
  - Limited number of clients permitted by V7.0 license without CAF

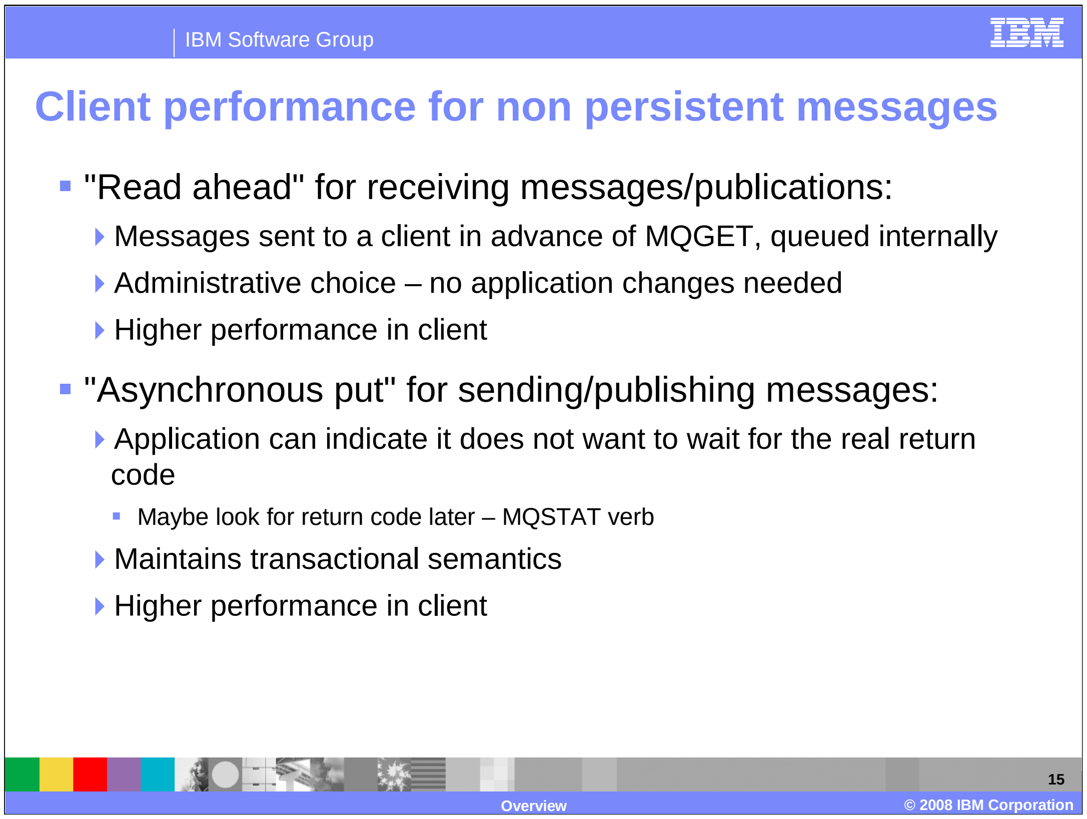
In earlier versions of WebSphere MQ, each client connection required a TCP/IP socket pair for its implementation.

In WebSphere MQ V7.0, when an application connects several times to the same queue manager, the connections are allowed to be shared over the same socket. This is especially useful for JMS applications that create several JMS sessions. In order to implement these features, client channels have been changed to use a full-duplex protocol (TCP/IP only). This implementation gives more heartbeat opportunities allowing for faster failure notification for clients and faster tidy-up of orphaned server-connection channels.

The client connection definition Table (CCDT) has always allowed a client to connect to one of several queue managers, but the list was always searched sequentially for the first available system. With WebSphere MQ V7.0, the selection can be configured to be made randomly from the list, assisting with workload distribution. Where multiple connections might be made from the same process, an affinity attribute tells the client to attempt to connect to the same queue manager as previously – This can then exploit the multiplexing capabilities of the channel.

The number of inbound client channels can now be restricted. You can specify independently the maximum number of instances of a channel, and the maximum number of instances from a specific partner. This is designed to prevent rogue applications from using all the system resources by continuing to connect. Existing attributes can still restrict the total number of channels across the queue manager.

With V7, the license for WebSphere MQ on z/OS has been modified to permit a small number of client connections to the queue manager without requiring purchase of the client attach facility. These connections are for administrative purposes only. If you want to have more connections or use clients for other purposes, then the CAF is still required.



Client performance for non persistent messages

- "Read ahead" for receiving messages/publications:
  - Messages sent to a client in advance of MQGET, queued internally
  - Administrative choice – no application changes needed
  - Higher performance in client
- "Asynchronous put" for sending/publishing messages:
  - Application can indicate it does not want to wait for the real return code
    - Maybe look for return code later – MQSTAT verb
  - Maintains transactional semantics
  - Higher performance in client

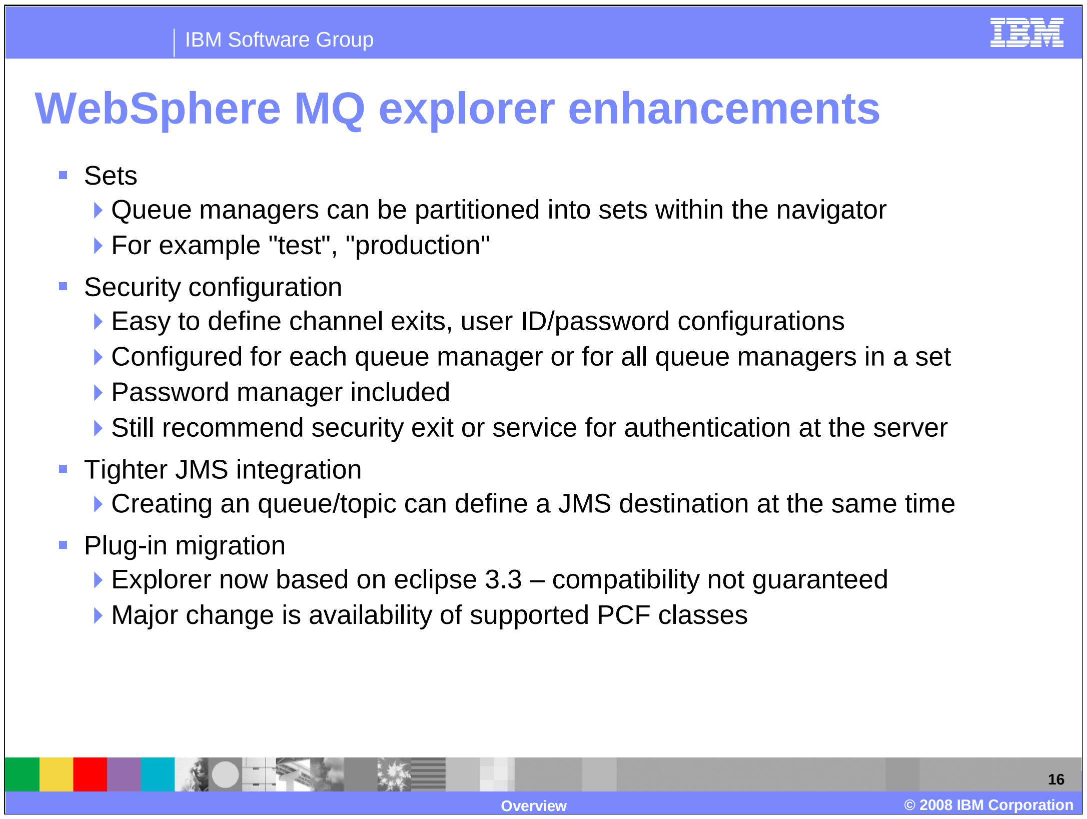
While WebSphere MQ non-persistent messages do not have assured delivery, they traditionally have still been more reliable than is necessary for some application scenarios. With V7.0 it is possible to have additional trade-off options for performance versus reliability. WebSphere MQ provides a comprehensive range of qualities of service for message delivery, ranging from assured delivery to lightweight non persistent.

Message read ahead is supported between MQ client code and servers, removing the need for the MQ client to specifically request every message that is sent to it by the server.

Messages are sent from the server to be held in a memory buffer on the client side. As an application program requests a message, it can often be supplied from the in memory buffer. As the buffer is consumed, the MQ client code will request additional messages from the server. Read ahead can provide a significant performance improvement to an application requiring a lower quality of service by removing many of the network replies.

Asynchronous put allows a client application to indicate that it is willing to send a sequence of MQPUT requests without each being individually acknowledged by the server. Failures of the MQPUT requests might not be notified until explicitly requested, or until a syncpoint is issued. This has the effect of reducing the number of network IOs required to deliver acknowledgements.

For both these options, both client code and server must be at version 7.0. If either side is at an earlier level the previous release behavior occurs.



WebSphere MQ explorer enhancements

- Sets
  - Queue managers can be partitioned into sets within the navigator
  - For example "test", "production"
- Security configuration
  - Easy to define channel exits, user ID/password configurations
  - Configured for each queue manager or for all queue managers in a set
  - Password manager included
  - Still recommend security exit or service for authentication at the server
- Tighter JMS integration
  - Creating an queue/topic can define a JMS destination at the same time
- Plug-in migration
  - Explorer now based on eclipse 3.3 – compatibility not guaranteed
  - Major change is availability of supported PCF classes

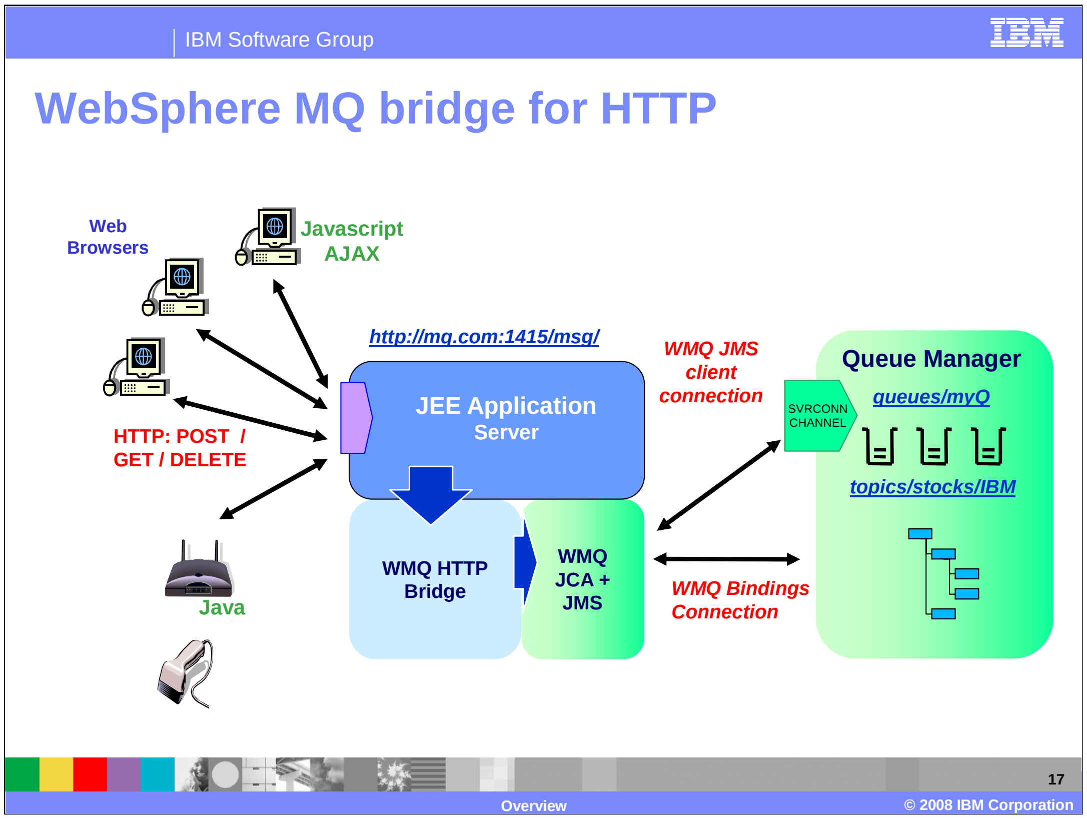
Along with support for new features, such as the publish/subscribe objects and their corresponding status displays, the explorer has been enhanced with more general capabilities.

One of the most frequently requested features has been the ability to partition the list of queue managers into sets, for example to show test and production systems separately in the navigation tree. This version of the explorer allows sets to be defined, with queue managers assigned to sets manually or automatically based on some attribute such as the CommandLevel value. Queue managers can appear in more than one set. There are some operations that can be applied at the set level, such as connecting to all of them with a single click.

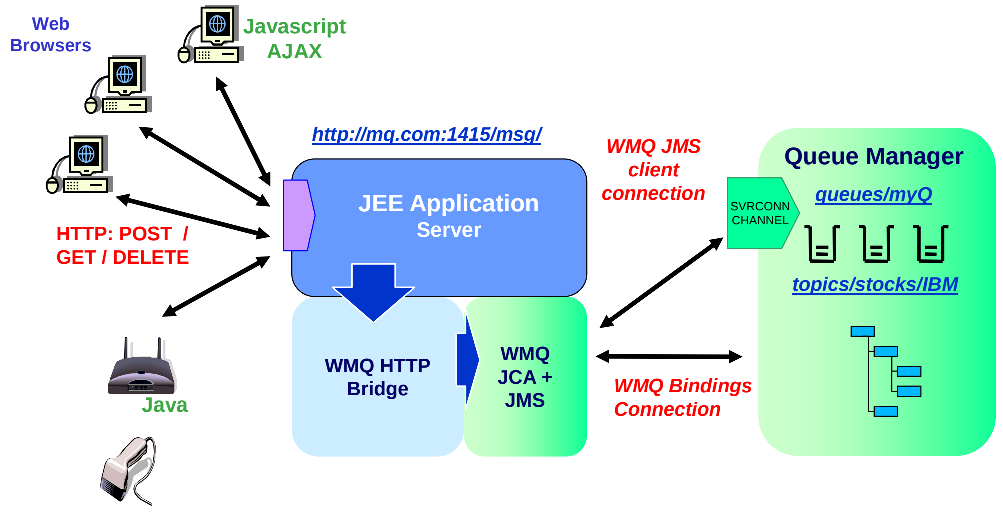
The security controls allow configuration of not just a user ID and password, but SSL and channel exit parameters. These can be defined globally or for each queue manager. A password manager component stores information securely, so that you do not need to re-enter passwords on each restart of the explorer.

When a queue or topic is created, the wizard can optionally also drive the corresponding JMS operation. This assist with integration of JMS configuration objects with the WebSphere MQ configuration.

The explorer has been updated to use the current version of the Eclipse runtime, v3.3. User-written plug-ins ought to work unchanged, but Eclipse has deprecated some interfaces that existed in the previous version, so compatibility cannot be guaranteed. One major change is that PCF classes are now officially available and supported; plug-ins should be changed to use those interfaces.



WebSphere MQ bridge for HTTP



The WebSphere MQ HTTP bridge is an enhanced version of the MA0Y support pack previously available. An alternative native implementation, SupportPac MA94 is also available as a separate download.

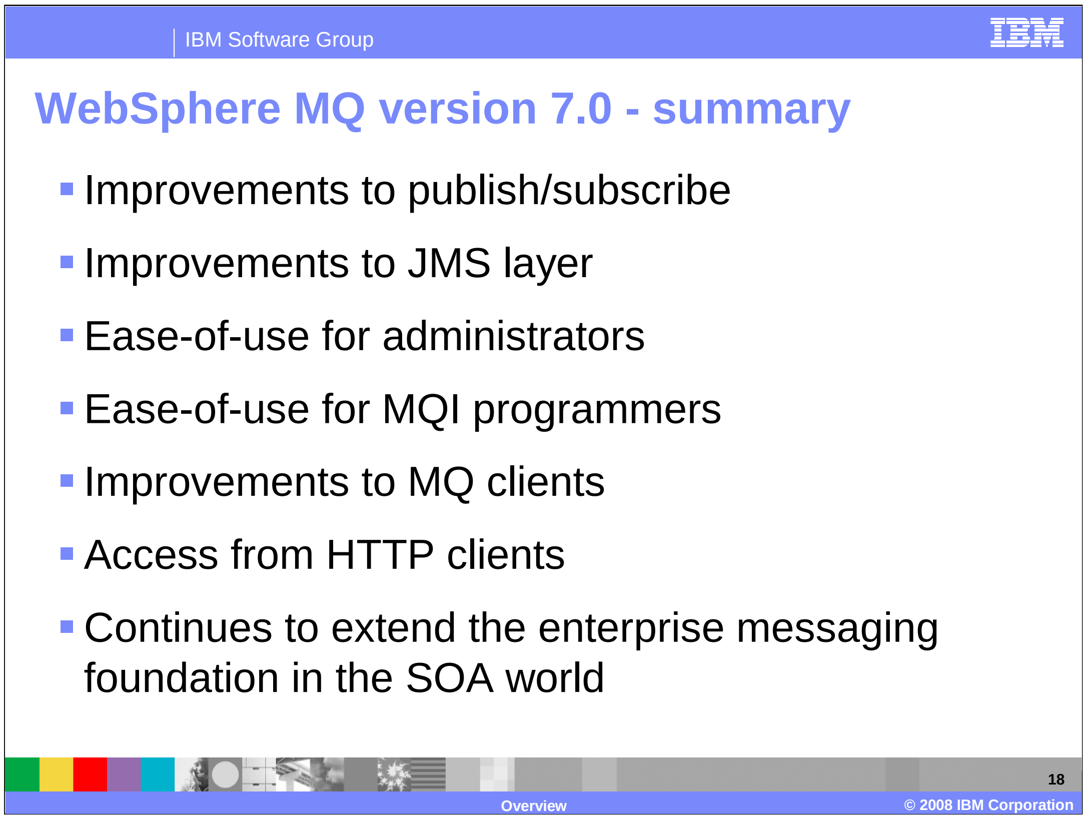
The main goal of the HTTP feature is to extend the reach of WebSphere MQ applications to more environments such as Web browsers. This will give rich internet applications simplified access to the enterprise. Eliminating the WebSphere MQ client reduces the cost of application deployment, though this is not a complete replacement for the WebSphere MQ client.

The API is modeled after REST ("representational state transfer") principles.

It is a stateless / connectionless API with one HTTP verb corresponding to one WebSphere MQ operation.

The queue or topic names are part of the URL of the HTTP request with options coded as HTTP headers.

No client libraries are required – Applications code directly to HTTP verbs using whatever APIs are in their environment. It is a genuinely zero footprint client.



WebSphere MQ version 7.0 - summary

- Improvements to publish/subscribe
- Improvements to JMS layer
- Ease-of-use for administrators
- Ease-of-use for MQI programmers
- Improvements to MQ clients
- Access from HTTP clients
- Continues to extend the enterprise messaging foundation in the SOA world

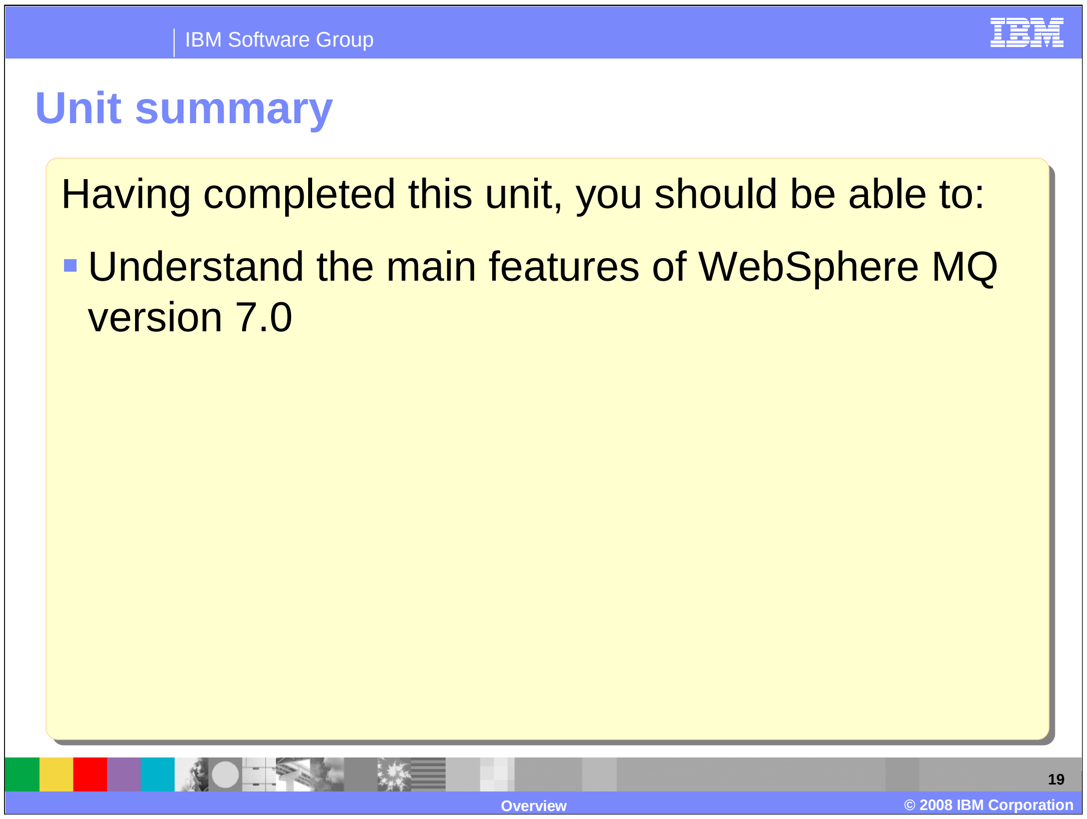
In summary, WebSphere MQ version 7.0 brings publish/subscribe functionality into the core of the product, extending the MQI and administrative interfaces to access topics and subscriptions in a familiar and easy to use manner.

The JMS implementation has been simplified to use the new features of MQ V7.0 directly.

MQ clients have been improved to give greater reliability and offer better performance for non persistent messages.

The reach of MQ has been extended to HTTP clients by the MQ HTTP bridge.

Overall, the release makes WebSphere MQ an even better JMS provider and makes many JMS features available to users of the MQI. It improves the experience of existing MQ client users and makes MQ available to a range of new HTTP clients.



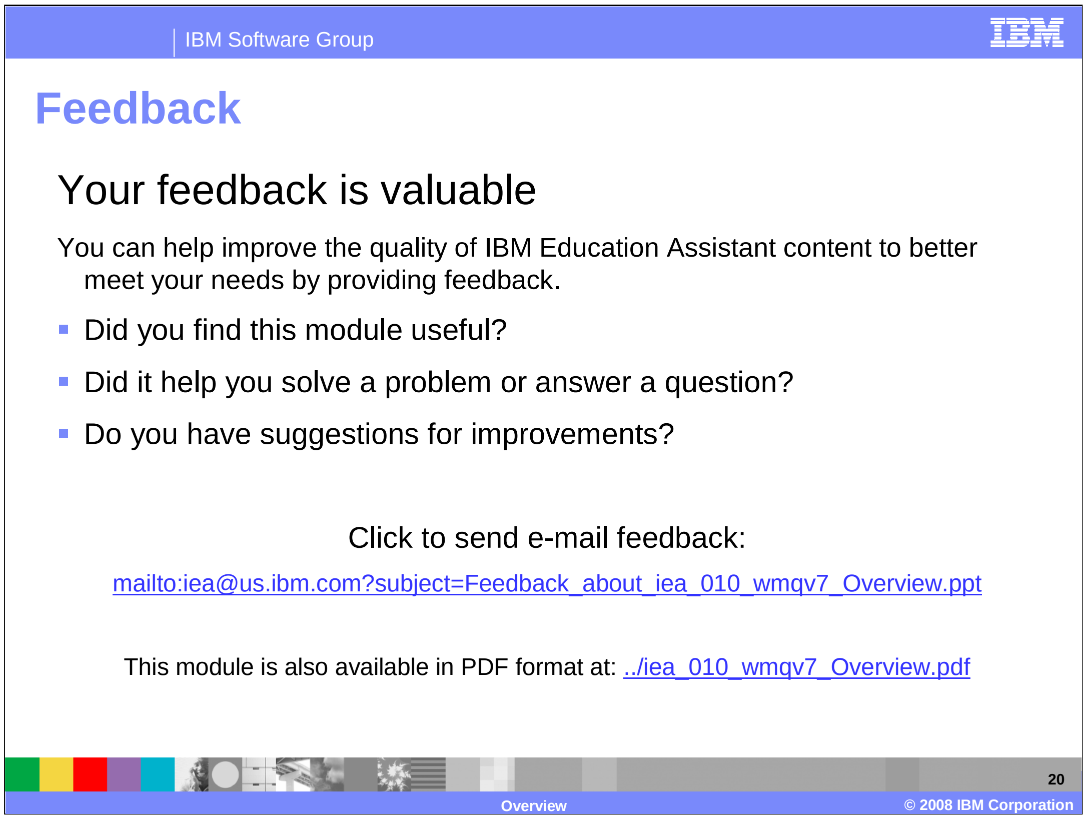
Unit summary

Having completed this unit, you should be able to:

- Understand the main features of WebSphere MQ version 7.0

This completes the overview of WebSphere MQ version 7.0 new features.

Additional presentations go into more detail for all these features and the product information center contains additional information.



Feedback

Your feedback is valuable

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

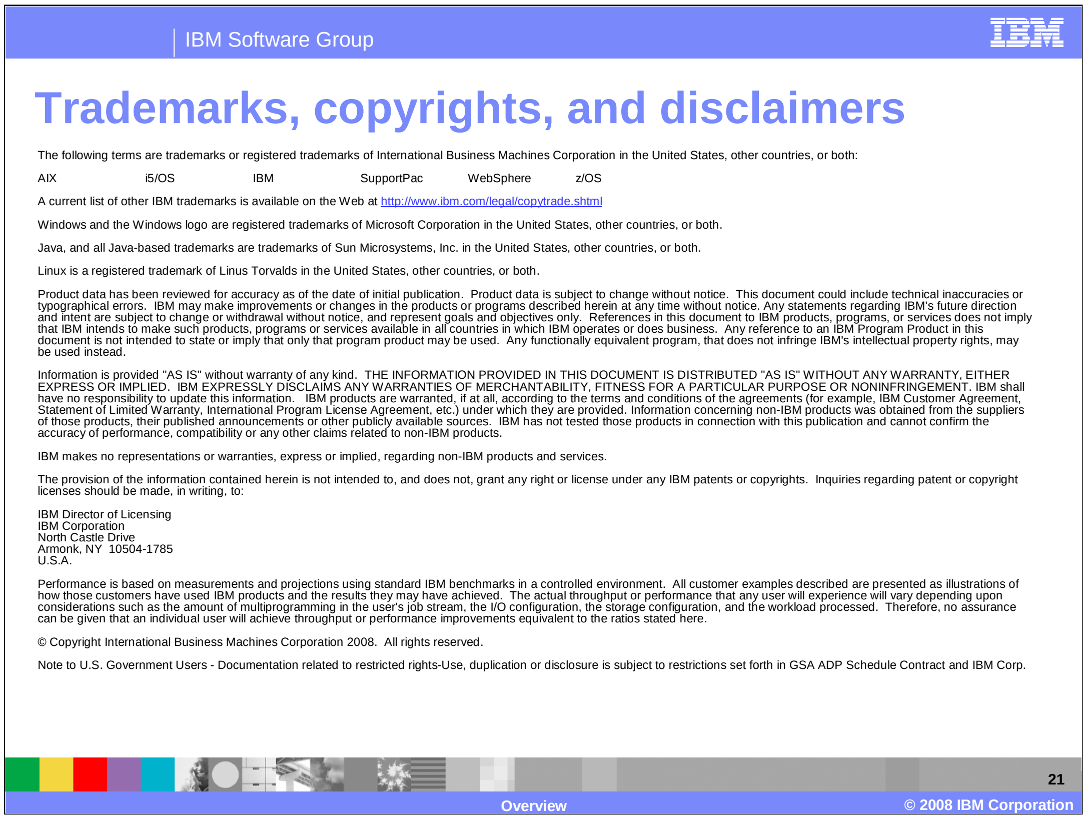
- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_iea_010_wmqv7_Overview.ppt

This module is also available in PDF format at: ../iea_010_wmqv7_Overview.pdf

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



Trademarks, copyrights, and disclaimers